J Supercomput (2015) 71:241–292
DOI 10.1007/s11227-014-1295-6

QRSF: QoS-aware resource scheduling framework in cloud computing

Sukhpal Singh · Inderveer Chana

Published online: 16 September 2014

Abstract Cloud computing harmonizes and delivers the ability of resource sharing over different geographical sites. Cloud resource scheduling is a tedious task due to the problem of finding the best match of resource-workload pair. The efficient management of dynamic nature of resource can be done with the help of cloud workloads. Till cloud workload is deliberated as a central capability, the resources cannot be utilized in an effective way. In literature, very few efficient resource scheduling policies for energy, cost and time constraint cloud workloads are reported. This paper presents an efficient cloud workload management framework in which cloud workloads have been identified, analyzed and clustered through K-means on the basis of weights assigned and their QoS requirements. Further scheduling has been done based on different scheduling policies and their corresponding algorithms. The performance of the proposed algorithms has been evaluated with existing scheduling policies through CloudSim toolkit. The experimental results show that the proposed framework gives better results in terms of energy consumption, execution cost and time of different cloud workloads as compared to existing algorithms.

Keywords Cloud workload · Cloud computing · Resource scheduling · Energy consumption · IaaS · Quality of service

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].

S. Singh (✉) · I. Chana
Computer Science and Engineering Department, Thapar University, Patiala, India
e-mail: ssgill@thapar.edu

I. Chana
e-mail: inderveer@thapar.edu

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 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 [2] and service level agreement (SLA) is an authorized agreement that describes QoS in written form [3]. 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, energy consumption and budget restrictions are met.

Resource scheduling is defined as the practice of implementing policies and procedures that improves the efficiency of computing resources in such a way so as to reduce the execution time and cost, energy consumption and environmental impact of their execution. 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 [4,5]. The mapping of workloads to appropriate resources for execution in cloud environment is a complex task and it can be easily solved using machine learning techniques. Through this technique, the classification of workloads based on their QoS requirements can be done. Dispersion, heterogeneity and uncertainty of resources bring challenges to resource allocation, which cannot be satisfied with traditional resource allocation policies in cloud [6]. 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 lead to maximization of scaling advantages. The amount of resources should be minimum for a workload to maintain a desirable level of QoS, or maximize throughput (or minimize workload completion time) of a workload. To address this problem, efficient solutions should be developed. To design a successful IaaS, initially understand the cloud workload (client-oriented, server-oriented and mobile-oriented) thoroughly based on QoS requirements [7]. Based on this, cloud consumer should design their applications which lead to maximization of the scaling advantage. With the help of this, not only dynamic infrastructure scaling can be achieved but also it will minimize the response time, execution cost, and energy consumption of elastic demand and maximize the throughput of requests [2]. A distinct workload (or a whole application) used by a set of consumers and a smaller facility may be used in different environments. The different applications have different set of QoS requirements and characteristics. Some clouds are natural fits for certain classes of workloads (i.e. web applications) whereas for another type of workloads (i.e. batch), other cloud services (AWS) are more necessary. It is difficult to prepare an IT resource to fulfill its processing desires. IT resources may be over-utilized or under-utilized depending on demand. The aim of workload analysis is to look at different aspects or characteristics of an enterprise application to determine the feasibility of moving or porting the application in the cloud.

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 cost of resources, (2) minimizing the execution time of workloads, (3) reducing the energy consumption and at the same time meeting the cloud workload deadline. The main aim of this research work is to: (1) propose cloud workload management framework, (2) clustering of workloads through machine learning techniques, (3) Propose 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], (4) optimize the execution cost and time for resource scheduling and simultaneously reduce the energy consumption, (5) perform evaluation with existing scheduling algorithms.

Paper is structured as follows: Sect. 2 presents related work and contributions. Cloud workload management framework (CWMF) with problem statement and objectives is presented in Sect. 3. Section 4 describes the experimental setup used for performance evaluation and results. A comprehensive comparison of CWMF with existing algorithms and experimental analysis of the performance of the proposed framework have been also described. Section 5 concludes the paper and Sect. 6 presents future directions.

2 Background and 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 cloud workload management systems in a cloud computing environment on the basis of resource requirements.

2.1 Cloud workload analysis

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. For example: running a web service or being a Hadoop data node are valid workloads. A workload is a self-governing services or group of code that can be implemented; workload does not depend on outside demands [8,9]. Nevertheless, most existing works have a slightly different emphasis or use a different method [10]. Different from these works, which emphasis on workload modeling at distinct server or general data center level, this study emphases on knowing the interrelated workload patterns within clusters of servers that result from application dependencies [10]. Moreover linked to this analysis are the works on capacity management and virtual server placement, which usually employ some workload modeling and forecast methods [11].

Cirne et al. [8] presented supercomputer workload model which addresses two aspects of job: probability of cancelation of job and requested time. In this model, pattern of job arrival and correlation between short execution time and poor request accuracy has been identified and considers only homogenous workloads. Arlitt et al. [12] identified, analyzed and classified the web servers based workloads. In this approach, different invariants of server based workloads have been identified to improve web server performance. Cherkasova et al. [10] conducted an analysis on broadcasting

servers and created a set of properties exclusive for enterprise broadcasting server's workloads. In this paper, a media server log analysis tool (MediaMetrics) has been proposed and defined metric rate of change to classify the workloads.

Gmach et al. [9] presented trace-based approach for capacity management and consider capacity planning and performance modeling to classify the workload demand patterns. Future demands based on the patterns have been predicted through the generation of synthetic workloads and analyzed the workloads to identify the nature of workloads. Bobroff et al. [13] proposed regression model based a dynamic server migration and consolidation technique to predict workload deviations, to dynamically place virtual machines (VMs). This approach decreases the amount of physical capacity necessary to provision a specified rate of SLA violations for a given workload and produced better results than static consolidation technique. Verma et al. [14] presented the first thorough investigation of an enterprise server workload from the viewpoint of finding features for consolidation. They focused on consolidating servers using association or peak cluster-based assignment when the correlation between applications is not considered. Rolia [11] described an automatic quartermaster capacity manager service for managing virtualization-based resource pools. A trace-based workload predicting technique was used for capacity management to improve performance and optimize search method.

These methods answer on the statistics (e.g., percentiles, peaks, etc.) of distinct workload time series to forecast upcoming capacity demand. In comparison, some other methods discover inter-server relationships and try to forecast workload levels at finer granularity. Khan et al. [15] described data traces acquired from a real data center to progress such abilities. Initially, they have searched for repeatable workload patterns by discovering cross-server performance relationships resulted from the dependencies among applications running on dissimilar servers. Chen et al. [16] inspected the performance aspects of virtualization standard based on the VMmark model which adapts the identical workloads on single server. Bossche et al. [17] studied this optimization problem in a multi-provider hybrid cloud setting with deadline-constrained and preemptible but non-provider migratable workloads that are categorized by memory, CPU and data transmission desires.

Xiong et al. [18] discussed two challenges: intrinsic optimization encounter between cost of resources and SLA agreement, and fluctuating demands of multiple tiers. They addressed these two open issues through the grouping of the resource controllers on both application and container levels to decrease the total amount of resources while meeting the E2E performance requirements for the workload. Tsakalozos et al. [19] suggested Nefeli, a virtual infrastructure gateway that is able to efficiently control different cloud workloads. This technique evades bottlenecks within the cloud through the recognition of feasible VM deployment mappings and presents a method to migrate VMs as desired to adjust to varying performance needs. Bossche et al. [20], handled this limitation by suggesting a set of procedures to cost efficiently schedule deadline-constrained bag-of-tasks applications on both public cloud providers and private infrastructure. Kousiouris et al. [21] predicted the influence of a number of critical factors on the performance of VMs. The critical factors considered in this research are allocation percentages, real-time scheduling decisions and migration of VMs.

Mahambre et al. [22] studied workload executing in infrastructure as a service (IaaS) cloud and classify into patterns, based on their behavioral features. They defined various kinds of behavioral patterns and outlined statistical methods to be used in defining these patterns. Here existing work of workloads in the context of cloud has been presented. They have not considered heterogeneous workloads. To develop an efficient resource scheduling framework, there is need to identify the heterogeneous cloud workloads and their QoS requirements.

2.2 Resource scheduling

Cloud computing is on demand as it offers dynamic flexible resource allocation for reliable and guaranteed services in pay according to use. Many cloud consumers can demand number of cloud services concurrently. Subsequently, there is a need to provide all the resources to request cloud consumer in a well-organized way to fulfill their requirements. Several optimized frameworks such as heterogeneous earliest time first (HEFT), ant colony optimization (ACO) based framework, resource aware scheduling algorithm (RASA) based framework, particle swarm optimization (PSO) based framework, optimal workflow based scheduling (OWS) framework among others have been proposed [23–27].

In a QoS constraint-based framework of cloud computing, QoS requirements are considered without violation of SLA. Varalakshmi et al. [26] described an OWS framework to discover a solution that tries to meet the user-desired QoS constraints, i.e. execution time. This paper shows slight improvement in resource utilization is attained. But it does not consider cost and energy as QoS parameters. Wang et al. [27] presented an 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. Xu et al. [28] presented a multiple QoS constrained scheduling framework for multiple workflows with different QoS requirements. This framework considered QoS requirement execution time only and not considered QoS requirements of resource consumers. Ambike et al. [29] proposed a non-preemptive priority queuing model based scheduling framework in which cloud user performs the activities and the QoS requirements are achieved. QoS based scheduling framework tries to negotiate with quality attributes and maximizes the performance and minimizes the execution cost by meeting the user requirements without considering energy consumption.

Yu et al. [30] proposed a cost-based workflow scheduling framework that reduces the only execution cost, however, meeting the deadline for delivering results. It can also adjust to the delays of service accomplishments by rescheduling unexecuted workloads. Sakellariou et al. [31] suggested a simple model for workflow applications that modeled as directed acyclic graph (DAG) and that permit to schedule the nodes of DAG onto resources in a method that fulfills a budget constraint and is optimized for overall time. Topcuoglu et al. [24] presented the 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 complete the workload at the earliest time. They did not design framework to reduce cost and time.

El-kenawy et al. [25] 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 RASA by considering only provider's benefit. Selvarani et al. [32] proposed a cost-based scheduling framework that divides all cloud consumer jobs depending on importance of every job into three different lists. The resource cost and computation performance have been measured and computation/communication ratio is improved. Dakshayini et al. [33] offered a priority-based scheduling framework that aims at serving the user requests satisfying the QoS, optimizing the time the service-request spends in the queue and achieving the high throughput of the cloud by making an efficient provision of cloud resources. They did not consider execution time and cost for independent parallel workload scheduling.

Ghanbari et al. [34] proposed a multi-criteria and multi-decision priority driven-based scheduling. The above two approaches generally devote a lot of time on fixing the priorities, so they are not appropriate for scheduling of heterogeneous cloud workloads. Wu et al. [35] suggested a market-oriented hierarchical scheduling approach which contains both service level scheduling and workload level scheduling. The service level scheduling deals with the task to service assignment and the workload level scheduling deals with optimization of the task to VM assignment in local cloud data centers. This framework was not considered cost and time as QoS parameters. Pandey et al. [23] introduced a PSO-based heuristic framework 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.

Ghorbannia et al. [36] proposed a trustworthy scheduling framework in cloud computing, wherein main problem is divided into sub problems (jobs). The request and acknowledge time are computed independently in the form of a shared job to make balance. Moschakis et al. [37] proposed a job migration and starvation handling based gang scheduling framework in which parallel jobs are scheduled. Initially, the system includes no VM, but depending on the computational needs of the jobs being serviced new VMs can be leased and later released dynamically. But it considers only cost as QoS parameter. Zhan et al. [38] presents PSO-based adaptive PSO framework to fulfill effective resource allocation in cloud environment at run time to improve the search efficiency and convergence speed but it does not considered cost and execution time. Liu et al. [39] presented workflow based compromised-time-cost scheduling framework which considers only execution time and cost simultaneously but did not consider energy consumption of resources. Verma et al. [40] 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.

All the above research works have presented scheduling frameworks in cloud computing without considering the execution time, cost and energy as QoS parameter for cloud environment and concept of rescheduling of cloud workloads to reduce number of deadlines missed. None of the existing work considers heterogeneous cloud workloads and their mapping to the appropriate resources based on clustering and four different types of resource scheduling policies simultaneously in a single cloud framework. Due to lack of negotiation between user and provider and use of conflicting resource scheduling policies, the communication time and cost can be increased. But when we use same resource scheduling policy then communication time and cost can be reduced and complexity can also be decreased. In addition, our novel cloud workload management framework needs to consider the basic features of cloud computing to execute the heterogeneous cloud workloads by negotiating makespan and budget with energy consumption which is not considered in other existing frameworks and the evaluation is performed by existing scheduling algorithms.

2.3 Our contributions

Our contribution in this research paper is twofold. Firstly, we have presented CWMF in which cloud workload analyzer (CWA) unit has been incorporated. CWA unit is responsible for identification, analysis and clustering of both homogenous and heterogeneous cloud workloads. Weights have been assigned on the basis of QoS requirements for each workload. The clustering of cloud workloads is done through K-means clustering algorithm as discussed in Sect. 3.5.3.2. Secondly, four resource scheduling policies [CCTB scheduling policy, time-based (TB) scheduling policy, cost-based (CB) scheduling policy and bargaining-based (BB) scheduling policy] are proposed based on different criteria. Mapping and execution of cloud workloads to the corresponding resources are done using these resource scheduling policies. We have considered execution time and cost of workload's execution as cloud consumer and cloud provider's functions, respectively. Decision tree-based scheduling criteria is used to select the scheduling policy based on the cloud consumer requirements. The proposed framework focuses on how to map the cloud workload to reduce cost, time and overall energy consumption using clustering. Finally, we have validated our proposed framework using CloudSim [41]. The main contribution of this paper is the development of CWMF that enables the mapping and execution of cloud workload according to cloud consumer requirements.

3 Cloud workload management framework

In cloud computing, resource scheduling is 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, cloud workloads are analyzed and clustered based on their requirements. In second step, identify the required set of resources from resource pool. In third step, map the cloud workload with appropriate resources based on QoS

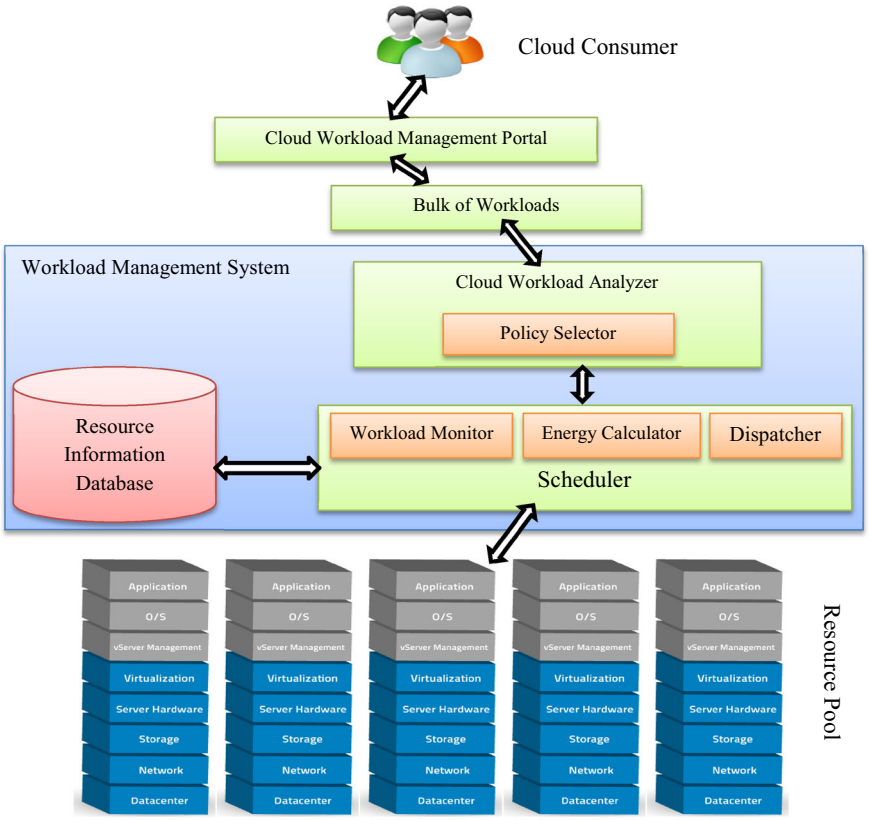

Fig. 1 Cloud workload management framework

requirements specified by user. In final step, schedule the workloads with user-specified resource scheduling policy therefore further guaranteeing near optimal satisfaction of QoS requirements. Need of optimized resource scheduling in IaaS can be achieved using the proposed framework. For example, assume that a customer wants to purchase some item from grocery store, then salesman would ask the range in terms of budget, 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. The proposed CWMF is shown in Fig. 1.

Flowchart shown in Fig. 2 depicts the flow of cloud workloads from CWMF to resource scheduling. The framework executes the requests as follows:

- In CWMF, first of all cloud consumer tries to execute the workloads through the cloud workload management portal (CWMP).
- After that, the task of cloud consumer's authorization and authentication is performed.
- After authentication, workload management system (WMS) asks to submit the cloud consumer requirements in the form of workload details, and authenticated

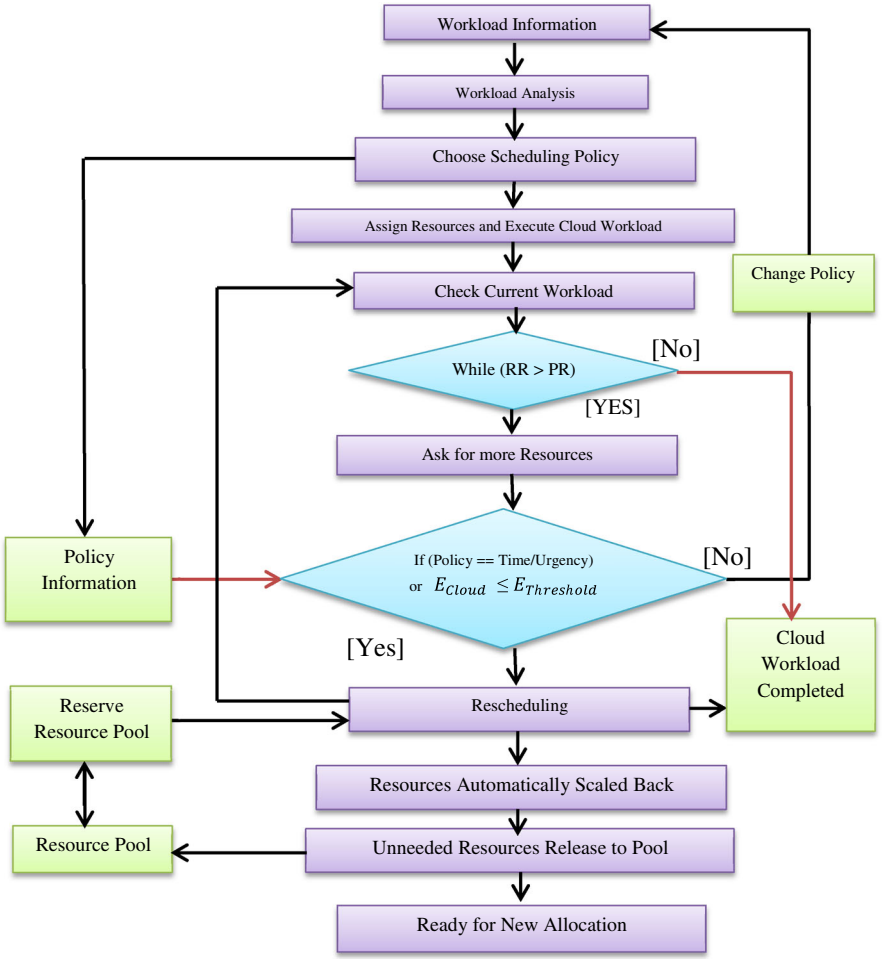

Fig. 2 Flowchart of cloud workload management framework

cloud consumer fills it and submits the request for the availability of particular resource with proper specification for the execution of their workload.

- WMS takes the information from the appropriate workload after analyzing the various workload details which cloud consumer demanded.
- WMS analyzes the workload and clusters them through K-means clustering algorithm.
- Decision tree is used to select the particular scheduling policy based on consumer workload details.
- WMS then collects the information available resources from resource information database (RID). RID contains details of all the resources available in resource pool and reserve resource pool.
- Based on cloud consumer details CWMF assigns resources and executes cloud workloads.

- During execution of a particular cloud workload, the CWMF will check the current workload. If the value of required resources (RR) is more than the value of provided resources (PR) then it will check the scheduling policy (See Sect. 3.5.5) in the following ways:
  - If the scheduling policy is time-based (TB) then it will reschedule the allocated resources to the cloud workloads, otherwise allocate the resources to new cloud workloads from reserve resource pool.
  - If the scheduling policy is cost-based (CB) then cloud workload management framework will ask to change the policy to execute cloud workloads and pay the amount as required.
  - If the scheduling policy is bargaining-based (BB) then the cloud workloads will be executed by negotiation or mutual agreement between cloud consumer and cloud provider.
  - If the scheduling policy is CCTB, cloud provider will minimize cost and execution time. For successful execution of a cloud workload, the actual energy consumption (E_{Cloud}) should also be less than threshold energy value ($E_{\text{Threshold}}$).
- After successful execution of cloud workloads, cloud workload management framework releases the free resources to resource pool and CWMF is ready for execution of new cloud workloads.

3.1 Objectives and commitments

The intent of CWMF is to ensure that the framework will map and execute the cloud workloads with QoS and user requirements. The following are main objectives of this work for better matchmaking and efficient scheduling:

(a) To clearly recognize the present and prospective upcoming requirements and expectations of the cloud customer.
(b) To analyze the cloud workloads and cluster them using appropriate machine learning algorithm.
(c) To reduce execution time and cost resulting in reduction of overall energy consumption.
(d) To improve user satisfaction by meeting customer requirements.

3.2 Framework assumptions

We have used some terms in this section: *Price* denotes the actual cost spent for execution of workload, *Urgency* denotes the requirement when user wants to execute their workload immediately in minimum time without reservation, *Stable* denotes that complete execution of workload from start to end without interruption and *Price List* denotes the set of prices of different resources based on user requirement. The proposed framework resulting from the following assumptions of cloud customer needs:

- Cloud consumer wants "price" to be less and under budget. A regular classification of price here is the quantity of expenses; total payment or cost incurred to finish all of the workloads under consideration.

- Cloud consumer wants "time" for completing all of the workloads to be less. If the dispatched workloads of a bulk of workloads (BoW) among a large number of resources, the latest completion time among all resources is the makespan of that workload and this completion time must be decreased as much as possible.
- Reducing the dynamic energy consumption by lowering the supply voltage at the cost of performance degradation. Develop resource management and scheduling algorithm that aim at minimizing the energy consumption and at the same time meet the cloud workload deadline too.
- Cloud customer prefers a "stable" workload assignment. Minimize unnecessary resource thrashing to minimize communication overheads.
- Cloud customer prefers an "urgency" workload assignment. Urgency refers to the "minimize" execution time of a particular assignment.
- This framework assumes a single IaaS provider with uniform price list of the resources being considered.
- If there is "urgency" then there is no need to move the workload into workload queue, processed directly using reserve resource pool.

3.3 Problem statement

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. $R = \{r_k 1 \leq k \leq n\}$ is the collection of resources and n is the total number of resources. $w = \{w_i | 1 \leq i \leq m\}$ 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, experimental data, cloud workload profiling and analytical benchmarking. For cloud workload management framework, the following constraints 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.

The list of symbols used in this research paper is described below in Table 1:

3.4 Objective function

In cloud computing, provider wants to minimize the execution time while user wants to minimize the cost for cloud workload. The goal of an objective function is to decrease the sum of product of cost and time expended for finishing all n workloads of a given BoW. This objective function (min z) 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:

$$\min z = \sum_{m=1}^{n} (E_t)_m \times (B_H)_m \tag{1}$$

where, m is a current workload that is being executed, E_t is execution time and B_H is budgeted per hour. The goal of cloud provider is to maximize the resource utilization and minimize the actual energy consumption.

The cloud workload will be executed only when the actual energy consumption (E_{Cloud}) is less than the threshold energy value ($E_{\text{Threshold}}$). 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 execution time and processor utilization is sufficient to measure the energy consumption for that cloud workload [42]. For a resource r_t at given time t, the utilization U_t is defined as (Eq. 2):

$$U_t = \sum_{b=1}^{c} U_{t,b} \tag{2}$$

Table 1 List of symbols

Notation	Description
Execution time (E_t)	Time required to execute the workload completely and measured in seconds
Budgeted per hour (B_H)	The amount of cost can spend in one hour for the execution of workload and measured in dollars ($)
Actual energy consumption (E_{Cloud})	The energy consumed for the execution of workload and measured in kilo watt hour (KWh)
Threshold energy value ($E_{\text{Threshold}}$)	The maximum value of energy consumption allowed for the execution of workload
Communication time (C_t)	Time required for communication between workload and resource during mapping and measured in seconds
Desired deadline (W_d)	The maximum time limit allowed to execute the workload as described by user and measured in seconds
Current time (Cur_t)	It denotes the present time and measured in seconds
Communication cost (C_c)	Amount of cost required for communication between workload and resource during mapping and measured in dollars ($)
Minimum cost ($C_{\min}$)	Minimum cost used to execute the workload and measured in dollars ($)
Estimated budget (B_E)	The maximum value of cost that user wants to spend and measured in dollars ($)
Resource price (P_r)	It denotes the price of single resource and measured in dollars ($)
Resource available (R_A)	Number of resources available in resource pool
Workload pending (W_p)	Number of workloads pending for execution
Available budget (B_A)	Budget available for the execution of a particular workload and measured in dollars ($)
Estimated completion time (ECT)	The approximate time used to complete the successful execution of workload and measured in seconds
Next schedule time (NST)	It denotes the next schedule of execution and measured in seconds
Total expected completion time (TECT)	The actual time required to complete the successful execution of workload. It is sum of execution time and communication time and measured in seconds
$\min z$	It denotes the sum of product of cost and time expended for finishing all n workloads of a given BoW
Total expected cost (TEC)	The actual cost required to complete the successful execution of workload. It is sum of minimum cost of execution and communication cost and measured in dollars ($)
Time difference (T_d)	It denotes the difference between the deadline time and total expected completion time and measured in seconds
Deadline time (D_t)	It is the difference between desired deadline and current time and measured in seconds
Deadline urgency (D_u)	It specifies cloud customer urgency to get workload (s) completed

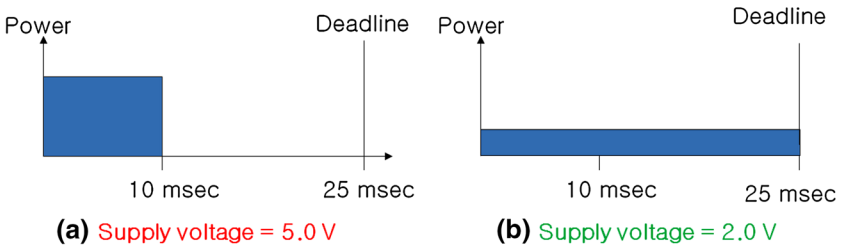

Fig. 3 Dynamic energy consumption

where c is the number of cloud workloads running at time t and $U_{t,b}$ is the resource usage of a cloud workload w_t [42]. The actual energy consumption E_{Cloud} of a resource r_t at given time t is defined as (Eq. 3):

$$E_{\text{Cloud}} = (\text{PC}_{\max} - \text{PC}_{\min}) \times U_a + \text{PC}_{\min} \tag{3}$$

where $\text{PC}_{\max}$ is the power consumption at the peak load (or 100 % utilization) and $\text{PC}_{\min}$ is the minimum power consumption in the active mode (or as low as 1 % utilization). Reducing the dynamic energy consumption by lowering the supply voltage at the cost of performance degradation. The supply voltage can be reduced when the more number of resources are in idle state. Suppose the deadline for workload execution is 25 ms and more number of resources are available than required, then execution can complete in 25 ms (supply voltage = 2.0 V) using lower number of resources rather than using more number of resources to complete it within 10 ms (supply voltage = 5.0 V). In idle state performance is degraded but it will not effect on the execution of workload and user satisfaction as shown in Fig. 3. Through this way, the workload can be executed within deadline with minimum energy consumption.

3.5 Framework units

The units of CWMF have been described as follows:

3.5.1 Cloud workload management portal

The cloud workload details are gathered through the CWMP from cloud consumer. Web browser acts as an interface for both consumer and provider. The cloud provider generates the workload schedule based on the workloads' details specified by the user. The workload generated by cloud provider is based on the four resource scheduling polices, to allocate the resources to the cloud workloads efficiently. Use case shown in Fig. 4 describes the core of the actual requirements of the CWMP.

3.5.2 Bulk of workloads

The number of cloud workloads submitted by the cloud user is processed in the queue. Based on the details given by user, the resources are assigned to the cloud workloads for their execution.

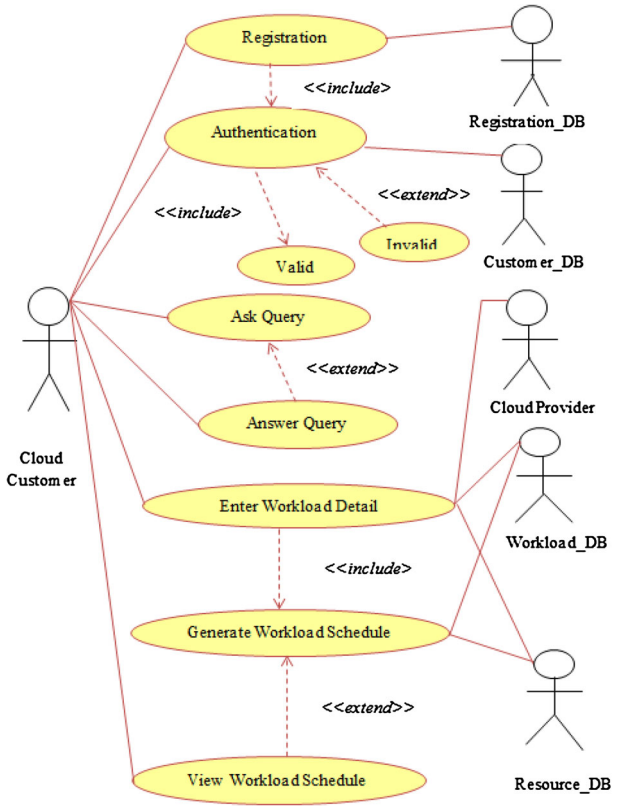

Fig. 4 Use case diagram of CWMP

3.5.3 Cloud workload analyzer

The aim of cloud 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 requirements and characteristics. This analysis also provides input to execution method, cloud service choice and a preliminary business worth valuation. The types of workload that have been identified during workload analysis [43], 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 [15,16,18,19,23,44,45]. After identification and analysis of workloads, they are classified on the basis of specific features in terms of security needs, network needs, variability of load, backup services, network bandwidth needs, computing capacity and other QoS metrics. Based on the QoS requirements of workload, Table 2 summarizes the classifications of cloud workloads.

Table 2 Classification of cloud workloads

Group	Workloads
Server oriented	Websites, technological computing, endeavour software, performance testing, online transaction processing, E-Com, central financial services, storage and backup services
Client oriented	Production applications, software/project development and testing, graphics oriented, critical internet applications
Mobile oriented	Mobile computing services

To schedule the resources efficiently, the clustering of cloud workloads is done. For clustering of cloud workload, assign weights to different quality attributes based on the importance for particular cloud workload.

3.5.3.1 Assign weights to quality attributes In this paper, the data (average of weights) collected from existing research papers from reputed journals are used because the researchers assign weights to quality attributes with respect to context in which that quality attribute is used. The range of weight scale has been assumed from 1 (minimum) to 5 (maximum). The weights are assigned according to the importance of a requirement for a particular cloud workload. If any quality attribute is not important for a particular cloud workload then zero or not available (NA) is assigned. These attributes are almost the same according to the international research. The weights for various quality attributes can be assigned from different types of research papers. Further, the average of every attribute has been taken and that average is the approximate weight (percentage) of that quality attributes [46] as specified in Eq. (4). The consequence of collected data is used by the following formula to calculate quality attributes weight:

$$W(i, j) = \frac{1}{N_f \times M_v} \times \sum_{k=1}^{N_f} R_k \times 100 \tag{4}$$

where in $W(i, j)$, i is cloud workload and j is quality attribute (QoS requirement) of that workload, N_f is number of research papers used to collect data, M_v is maximum value for a quality attribute and R_k is response for an attribute; the value of $W(i, j)$ will be in the range 0–100 %. An analysis has been conducted to acquire the data from 15 research papers of cloud computing from reputed journals about cloud workloads with the objective to know about how to assign the weights to the quality attributes according to significance [47–61]. After getting the responses, an industry standard baseline and acceptable weights to the quality attributes have been defined. The conversion metric is used to assign the values (minimum = 1 and maximum = 5) [62] corresponding to the percentage as shown in Table 3.

This information determines the authenticity of the data that is received from different research papers. The result of the data analysis is as follows; a total 15 research papers of different contexts have been studied and maximum possible value for an

Table 3 Conversion metric

Approximate weight (%)	Weight
0–20	1
20–40	2
40–60	3
60–80	4
80–100	5

attribute is 5. For example calculating the average of "usability" quality attribute by putting the values in Eq. (4) is as following:

$N_f = 12$, $i =$ online transaction processing and $j=$ usability, $M_v = 5$ and sum of the responses $\sum_{k=1}^{12} R_k = 32$

$$W(i, j) = \frac{1}{12 \times 5} \times 32 \times 100 = 53.33$$

For $W(i, j) = 53.33$, the average weight is assigned for usability is 3 using Table 3. Through this the average weights for every quality attribute have been calculated.

3.5.3.2 Cloud workload based K-means algorithm for clustering of workloads As workload demands vary widely and are quite fluctuating simple static resource provisioning results in over and under provisioning. cloud workload K-means clustering algorithm is non-hierarchical method that initially takes the number of workloads of the population [workload set (W)] equals to the final necessary number of clusters. The actual essential number of clusters is selected such that the workloads are mutually farthest apart based on QoS requirements in this step. Next, it examines each workload in the population and assigns it to one of the clusters depending on the minimum distance. The cluster centroid's (C) position is recalculated every time a workload is added to the cluster and this continues until all the workloads are grouped into the final required number of clusters (C_n) [63]. The algorithm used for clustering of cloud workload is shown in Fig. 5. Cloud workload K-means clustering algorithm calculates the distance between each workloads and select that pair which shows the minimum distance and remove it from actual workload set (W). Then take one workload from workload set ($W = \{W_1 W_2 \ldots W_t\}$) and calculate the distance between selected workload and standard workload from workload set (W) and add with that cluster which shows the minimum distance. Repeat this process till threshold value is achieved. If number of value is less than k then again calculate the distance between each workload from the rest workload set (W) and repeat that process till k cluster is formed.

3.5.3.3 Weight assignment for cloud workloads The range of weight scale has been assumed from 1 (minimum) to 5 (maximum). The weights are assigned according to the importance of a requirement for a particular cloud workload as shown in Table 4.

Abbreviations for the workload and their corresponding requirements have been presented in Table 5.

Based on the importance of a requirement for a particular cloud workload the data values have been assigned as shown in Table 6.

Input:

W = {$W_1 W_2$ ……W_t } (Workloads to be clustred)

C_n = number of clusters

Output:

C = {$C_1 C_2$ ……C_m } (Cluster Centroid)

m:w → {1, 2, 3……m} (Cluster Membership)

Procedure: K means

Set C to initial value (Random selection of workload (w))

For each W_x ϵ W

m (W_x) = argMinDistance (W_x , C_y)

y ϵ{1…….m}

End

While m has changed value

For each x ϵ {1, 2, 3…….m}

Recompute C_x as the centroid {w|m (w) = x}

End

For each W_x ϵ W

m (W_x) = argMinDistance (W_x , C_y)

y ϵ{1…….m}

End

End

End

Fig. 5 *K*-Means algorithm for clustering of cloud workloads

Let the four seeds [64] be the four workloads as shown in Table 7.

Now compute the distance using the four values (weights assigned) and using the sum of differences for simplicity (i.e. using the *K*-median method [65]). The distance values for all the cloud workloads are given in Table 8, wherein columns 6, 7, 8 and 9 give the four distances from four seeds, respectively. Based on these distances workload is allocated to the nearest cluster [66] and the result of first iteration as shown in Table 8.

First iteration leads to two each workload in first and second cluster, three in third cluster and six in fourth cluster. Table 9 compares [67] the cluster means of cluster found in Table 8 with the original seeds (s1, s2, s3, s4).

Use the new cluster means to recompute the distance of each object to each of the means, again allocating each cloud workload to the adjacent cluster. Table 10 shows the second iteration.

The number of workloads in all the four clusters is again same. A more careful look shows that the clusters have not changed at all. The cluster membership [64] is shown in Table 11.

3.5.4 Resource information database

The resource details include the number of CPU use, size of memory, cost of resources, type of resources and number of resources. All the common resources are stored in resource pool and reserve pool contains some reserve resources.

Table 4 Workloads with their requirements and weights

Id	Workload	QoS requirements	Weights assigned
W1	Web sites	Reliable storage	3
		High network bandwidth	3
		High availability	5
W2	Technological computing	Computing capacity	5
W3	Endeavour software	Security	5
		High availability	5
		Customer confidence level	3
		Correctness	3
W4	Performance testing	Computing capacity	5
W5	Online transaction processing	Security	5
		High availability,	3
		Internet accessibility	5
		Usability	3
W6	E-Com	Variable computing load	5
		Customizability	3
W7	Central financial services	Security	5
		High availability	3
		Changeability	1
		Integrity	5
W8	Storage and backup services	Reliability	5
		Persistence	3
W9	Productivity applications	Network bandwidth	2
		Latency	3
		Data backup	4
		Security	5
W10	Software/project development and testing	User self-service rate	4
		Flexibility	4
		Creative group of infrastructure services	1
		Testing time	5
W11	Graphics oriented	Network bandwidth	3
		Latency	3
		Data backup	5
		Visibility	4
W12	Critical internet applications	High availability	5
		Serviceability	4
		Usability	3
W13	Mobile computing services	High availability	3
		Reliability	5
		Portability	2

Table 5 Abbreviations of workload requirements

Network bandwidth	R1	Variable computing load	R13
Integrity	R2	User self service rate	R14
Security	R3	Reliable storage	R15
Usability	R4	Database backup	R16
Reliability	R5	Correctness	R17
Availability	R6	Visibility	R18
Changeability	R7	Serviceability	R19
Latency	R8	Computing capacity	R20
Customer confidence level	R9	Flexibility	R21
Portability	R10	Internet accessibility	R22
Customizability	R11	Persistence	R23
Testing time	R12	Creative group of infrastructure services	R24

Table 6 Data values for workloads

Workloads	Requirements	Value 1	Value 2	Value 3	Value 4
W1	R15, R1, R6	3	3	5	0
W2	R20	5	0	0	0
W3	R3, R6, R9, R17	5	5	3	3
W4	R20	5	0	0	0
W5	R3, R6, R22, R4	5	3	5	3
W6	R13, R11	5	3	0	0
W7	R3, R6, R7, R2	5	3	1	5
W8	R5, R23	5	3	0	0
W9	R16, R1, R3, R8	2	3	4	5
W10	R14, R21, R24, R12	4	4	1	5
W11	R1, R8, R16, R18	3	3	5	4
W12	R6, R19, R4	5	4	3	0
W13	R5, R6, R10	3	5	2	0

Table 7 The four seeds for given workloads

Seed	Value 1 (V1)	Value 2 (V2)	Value 3 (V3)	Value 4 (V4)
s1	5	0	0	0
s2	5	3	0	0
s3	3	3	5	0
s4	5	3	5	3

3.5.5 Policy selector

Four resource scheduling policies [CCTB scheduling policy, time-based (TB) scheduling policy, cost-based (CB) scheduling policy and bargaining-based (BB) scheduling policy] are proposed in this paper. Decision tree is used to select the appropriate pol-

Table 8 First iteration: allocating each cloud workload to the nearest cluster

C1	5	0	0	0	Distance from clusters				Allocation to the nearest cluster
C2	5	3	0	0	Distance from C1	Distance from C2	Distance from C3	Distance from C4	
C3	3	3	5	0					
C4	5	3	5	3					
Workload	V1	V2	V3	V4					
W1	3	3	5	0	6	3	0	5	C3
W2	5	0	0	0	0	3	6	11	C1
W3	5	5	3	3	11	8	5	0	C4
W4	5	0	0	0	0	3	6	11	C1
W5	5	3	5	3	11	8	5	0	C4
W6	5	3	0	0	3	0	3	8	C2
W7	5	3	1	5	9	6	3	2	C4
W8	5	3	0	0	3	0	3	8	C2
W9	2	3	4	5	9	6	3	2	C4
W10	4	4	1	5	9	6	3	2	C4
W11	3	3	5	4	10	7	4	1	C4
W12	5	4	3	0	7	4	1	4	C3
W13	3	5	2	0	5	2	1	5	C3

Table 9 Comparing new centroids and the seeds

	Value 1	Value 2	Value 3	Value 4
C1	5	0	0	0
C2	5	3	0	0
C3	3.6	4	3.3	0
C4	4	3.5	3.1	4.1
s1	5	0	0	0
s2	5	3	0	0
s3	3	3	5	0
s4	5	3	5	3

icy based on workload details described by cloud consumer. Cloud environment and a scheduler that implements different scheduling policies based on the decision taken by cloud provider. Resource scheduling procedure is shown in Fig. 6. Based on the scheduling policy, the resources are allocated to the cloud workloads. The information of the cloud workloads and computational resources is send to the allocation agent. The allocation agent implements four resource scheduling policies: CCTB, TB, CB and BB scheduling policy. Cloud workload management portal 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

Table 10 Second iteration: allocating each cloud workload to the nearest cluster

C1	5	0	0	0	Distance from clusters				Allocation to the nearest cluster
C2	5	3	0	0	Distance from C1	Distance from C2	Distance from C3	Distance from C4	
C3	3.6	4	3.3	0					
C4	4	3.5	3.1	4.1					
Workload	V1	V2	V3	V4					
W1	3	3	5	0	6	3	0.1	3.7	C3
W2	5	0	0	0	0	3	5.9	9.7	C1
W3	5	5	3	3	11	8	5.1	1.3	C4
W4	5	0	0	0	0	3	5.9	9.7	C1
W5	5	3	5	3	11	8	5.1	1.3	C4
W6	5	3	0	0	3	0	2.9	6.7	C2
W7	5	3	1	5	9	6	3.1	0.7	C4
W8	5	3	0	0	3	0	2.9	6.7	C2
W9	2	3	4	5	9	6	3.1	0.7	C4
W10	4	4	1	5	9	6	3.1	0.7	C4
W11	3	3	5	4	10	7	4.1	0.3	C4
W12	5	4	3	0	7	4	1.1	2.7	C3
W13	3	5	2	0	5	2	0.1	4.7	C3

Table 11 Cluster membership

Cluster (C_n)	Cluster name	Workloads
C1	Compute	W2, W4
C2	Storage	W6, W8
C3	Communication	W1, W12, W13
C4	Administration	W3, W5, W7, W9, W10, W11

represented in the scheduling request of the cloud workload. Similarly, QoS, such as processing speed, is generated for each computational resource.

3.5.5.1 Cloud workload attributes 3,000 independent cloud workloads were generated randomly in CloudSim as cloudlets [41]. For each cloud workload, the attributes of the cloud workload include deadline, estimated budget and scheduling policy. The resource scheduling policies consider only one-dimensional QoS (processing speed). The QoS is generated to be 1–32 with the ratio following given distribution of the QoS request. In this paper, the six scenarios of the QoS distribution have been defined (See Table 14).

3.5.5.2 Resource attributes For each set of resources, the attributes of the resource include number of resources (or computing nodes), QoS provided, and the information

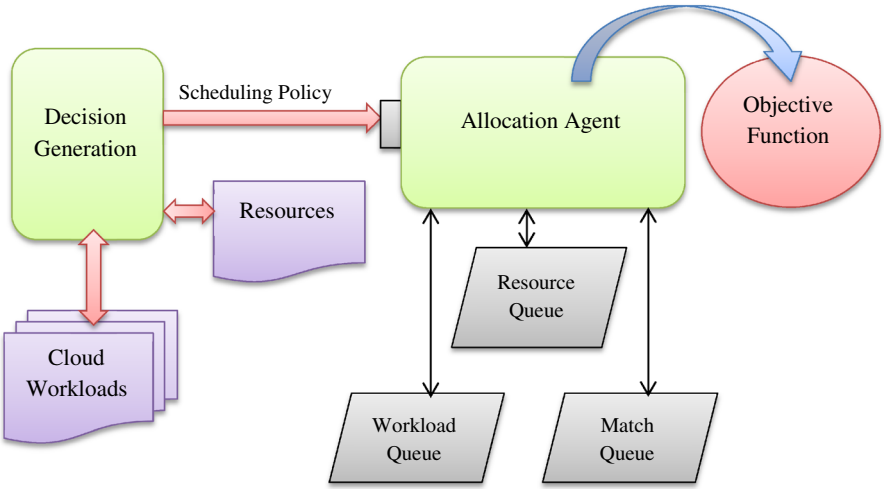

Fig. 6 Resource scheduling procedure

of each resource, which includes the deadline, estimated budget and scheduling policy. All configurations about the resources will remain the same during the experiment. In this paper, one-dimensional QoS (processing speed) has been implemented.

3.5.5.3 Allocation agent The allocation agent receives the cloud workloads and puts them into the workload queue. While the workload queue is not empty, the allocation starts the scheduling policy to find the right workload-resource match according to the policy. To compare proposed QoS guided policies with the existing scheduling policies, all the proposed policies have been implemented to get the performance data.

3.5.5.4 Decision tree-based scheduling criteria Classification tree analysis is used when the predicted outcome is the class to which the data belong. Regression tree analysis is used when the predicted outcome can be considered as a real number.

The classification and regression objective [68] require 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 12). Each object is characterized with 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 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:

Table 12 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

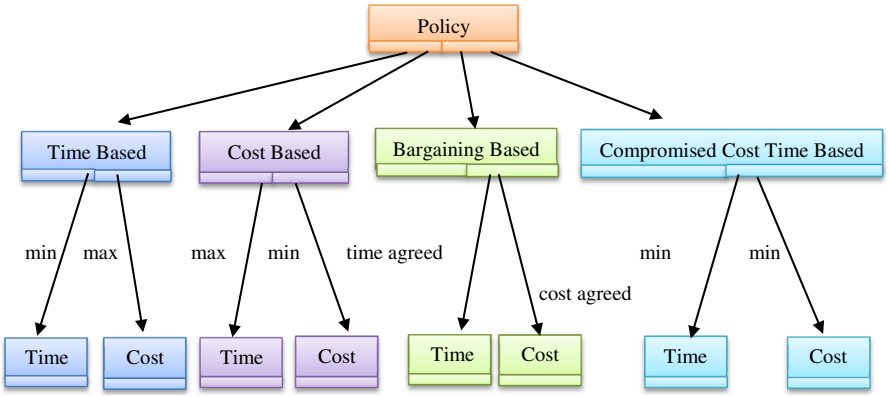

Fig. 7 Decision tree of scheduling policies

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

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 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 12.

Decision trees comprise a way of presenting rules in a hierarchical, sequential structure. Figure 7 shows a decision tree for the data presented in Table 12. There are different ways to transform logical and categorical variables into numeric ones. Logical types, as a rule, are encoded by the figures 1 and 0.

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}.

Figure 7 shows the selection of policy based on the workload details provided by user through the use of Table 12.

Policy: Compromised Cost-Time Based (CCTB) Scheduling Policy

Data:
Name of Workload
Type of workload
Desired Deadline (W_d)
Estimated Budget (B_E)
Preferred Policy = CCTB
Result:
Each workload will be mapped to the resources within available budget and desired deadline as per the specified policy.
Begin:
Intilize resourceList [No. of Resources]
Initialize workloadList [No. of Workloads]
resourceList = getAvailbleResource()
workloadList = getWorkloadtoSchedule()
Step 1:
a) Group Workloads into two categories:
- Homogenous Workload
- Heterogeneous Workload

Homogenous workloads based on QoS as well as some relationship among them (Dependency of workload (s) to another workload (s))
b) Evaluate the minimum execution time and price for every workload from the available set of resources by using cost and time fitness formulas.
c) Calculate TECT
d) Calculate D_t
e) Calculate TEC
f) If (TECT ≤ D_t && TEC ≤ B_E)
{
If (E_{Cloud} ≤ $E_{Threshold}$)
{
Dispatch the workload
}
else
{
Redistribute workloads
}
}
else
{
Change the Scheduling Policy
}
Step 2:
a) Allot the Cloud customer whole deadline and budget into every workload partition in proportion to their least execution time and cost respectively calculated in step 1 (b).
The deadline and budget is spread according to following rule: Execution time of workload may differ. Some workload may require only small time to be completed and some need at least more than 2 hours.
There are many probable execution time and cost for each workload but use only E_t and C_{min} to allocate the overall deadline and given budget correspondingly.
b) Sorting the entire resource list by giving highest priority to the expensive one based on their cost.
Step 3:
Select a resource to process a specific workload from the workload list so that (E_t < W_d && TEC ≤ B_E).
Step 4:
Repeat the step 3 until all the Cloud workloads within both the partition have been scheduled and executed, otherwise rescheduling the Cloud workloads to the resources.

Fig. 8 Compromised cost-time based (CCTB) scheduling policy

3.5.5.5 Compromised cost-time based (CCTB) scheduling policy In this scheduling policy, cloud provider minimizes cost as well as execution time along with least energy consumption. It calculates the total expected cost (TEC), total expected completion time (TECT) and time difference (T_d) to allocate the resources as specified in Eqs. (5), (6) and (7). The allocation agent finds the missed deadlines and calculates Time Difference for each workload then uses the extra available time to the workloads with missed deadlines and executes all the cloud workloads within their corresponding deadlines. The CCTB scheduling policy is shown in Fig. 8. The fitness value (TECT, TEC, D_t) is calculated as follows:

$$\text{Total expected completion time (TECT)} = C_t + E_t \tag{5}$$

$$\text{Deadline time } (D_t) = D_t = W_d - \text{Cur}_t \tag{6}$$

$$\text{Total expected cost (TEC)} = C_c + C_{\min} \tag{7}$$

The complexity (Eq. 8) of CCTB scheduling policy is influenced by number of change points (CP), i.e. rescheduling and requested resources (r) of workload being scheduled. Here,

$$\text{CP} = (S_t + T_e + T_s + R_t) \tag{8}$$

where, S_t = start times of all workloads, T_e = end times of all workloads, T_s = suspend times of all workloads, R_t = resume times of all workloads.

Consider lesser pre-emption as its objective. The complexity of the algorithm mainly depends on two important objectives:

- Minimize the rejection rate of the incoming requests.
- Minimize reshuffle cost (avoid rescheduling of already accommodated leases as much as possible).

The mapping is done with the objective of minimum cost, time and energy simultaneously.

3.5.5.6 Cost-based (CB) scheduling policy Cost-based (CB) scheduling policy works as per following: first, the allocation agent begins to compute the cost of each cloud workload then sort, as the priority is given to the cloud workload which has maximum budget. If the two workloads have same budget then that workload will execute first 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 cost-based (CB) scheduling policy is shown in Fig. 9. The fitness value (Estimated Cost) is calculated as follows (Eq. 9):

$$\text{Estimated cost: budgeted/hour } (B_H) = \frac{B_E}{W_d - \text{Cur}_t} \tag{9}$$

The mapping is done with the objective of minimum cost for workload execution. To maximize the chance that the desired deadline can still be met after terminating one resource, termination is only done if the estimated completion time is lesser than a desired deadline ($E_t < W_d$). In the current implementation of cost-based scheduling policy, consider as a constant coefficient.

3.5.5.7 Time-based scheduling policy Time-based (TB) scheduling policy works as per 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 shortest deadline time (D_t) will execute first. If the two workloads have same deadline time then the workload will execute first that has lesser execution time. By default, PS = 1. The allocation agent then schedules all the cloud workloads with smallest execution time request to the resources that provide high QoS. If any deadline found missed then recalculate the execution time by increasing the value of processing

Policy: Cost Based (CB) Scheduling Policy

Input: W_d , B_E , P_r

1. Set B_A (B_E)
2. While W_p () = True do
3. If ($R_A > 0$)
4. AdmitWorkload (R_A)
5. E_t = ECT
6. *else*
7. $E_t = W_d + 1$
8. If ($E_t > W_d$) then
9. $B_A = Get\ B_A$ ()
10. *If* ($B_A \geq P_r$) then
11. $AskResource(1)$
12. $R_A = R_A + AddResources()$ //Available Resource Pool
13. DoMapping()
14. else
15. If ($E_t < W_d$) then
16. Finish (1)

Fig. 9 Cost-based (CB) scheduling policy

Policy: Time Based (TB) Scheduling Policy

Input: W_d , B_E , P_r

1. Calculate B_H

$$B_H = \frac{B_E}{W_d - Cur_t}$$

2. $askCount = \frac{B_H}{P_r}$
3. askResource (askCount)
4. if (Criteria == Urgency)

{

$R_A = R_A + AddReserveResources()$ //Reserve Resource Pool

}

else

{

$R_A = R_A + AddResources()$ //Available Resource Pool

}

5. DoMapping based on Criteria
6. While W_p ( ) = True do
7. AdmitWorkload (R_A)
8. Finish (askCount)

Fig. 10 Time-based (TB) scheduling policy

speed (PS) and it will increase cost only. The time-based (TB) scheduling policy is shown in Fig. 10. The fitness value (estimated time) is calculated as follows (Eqs. 10, 11):

For homogenous workloads

$$\text{Execution time } (E_t) = \text{Workload remained } * \text{ Workload runtime} \tag{10}$$

For heterogeneous workloads

$$\text{Execution time } (E_t) = \sum_{i=1}^{n} W_i \text{ Runtime} \tag{11}$$

DoMapping(), in both cost and time scheduling policies, takes care of budgeting for hired resources and decreases the available budget based on the price of the hired

Policy: Bargaining Based (BB) Scheduling Policy

```
1.  MappingList ←empty
2.  While Cur_t ≤ NST do
3.          IncomingResource (A_y) // From Cloud Providers
4.          IncomingWorkload (W_y) // From Cloud Customers
5.  end while
6.  CalculateAvailable(W_y)
7.  CalculateRequirement(A_y)
8.  UpdateGivenCost (W_y)
9.  UpdateTakenCost (A_y)
10. ListTaken ← Sort_Taken(A_y)
11. ListGiven ← Sort_Given(A_y)
12. Let y=0 and s= avilableQueueSlots(t)
13. for every Given x in the list Given do
14.     g ← given(x) for every workload y of Given g do
15.       If (s =! 0) then
16.         If t.value() =! g.value() then
17.           If Check_Desired_Deadline (x, t)
18.             mapping y = AllocateResource (x, t)
19.             addingMapingList (Map_List)
20.             s--
21.           else
22.             Goto line number 36
23.           endif
24.         else
25.           If IsEmpty (workloads) then
26.             break
27.           else
28.             y++
29.             t ← taken(y)
30.           endif
31.         endif
32.       else
33. endif
34.     end for every
35. end for every
36. for each instance in Map_List do
37. IntimateCloudCustomer()
38. end for every
```

Fig. 11 Bargaining based (BB) scheduling policy

resources per hour. If there is not enough budget, then DoMapping() terminates each hired resource before it starts a new mapping cycle [69]. The mapping is done with the objective of minimum execution time for workload execution.

3.5.5.8 Bargaining-based (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 bargaining-based (BB) scheduling policy is shown in Fig. 11. The mapping is done with the objective of best negotiation between consumer and provider. The fitness value (deadline urgency) is calculated as follows:

- Deadline urgency—Deadline urgency (D_u), which specifies cloud customer urgency to get workload (s) completed, is defined as (Eq. 12):

$$D_u = \frac{[W_d - S_t]}{[E_t]} - 1 \tag{12}$$

where S_t is the start time of the user workload, W_d is desired deadline and E_t is the execution time of cloud customer's workload. The deadline is considered very urgent when $D_u < 0.25$, intermediate when $0.25 < D_u < 0.75$ and relaxed when $D_u > 0.75$. This metric shows how the scheduler deals with cloud customer with different requirements on time.

- Budget per workload: The budget provided by the cloud customer for their workload is divided by the number of workloads contained within the application to normalize the budget across all the workloads. This metric examines how the schedulers allocate resources fairly among different cloud customers with different budget groups.
- Amount of deadlines lost with rise in quantity of cloud customer workloads: This metric is used to examine how the scheduling algorithms are able to cope up with multiple cloud customers when requirement for resources exceeds their availability.

3.5.6 Scheduler

Scheduler is used to schedule the cloud workloads and map the cloud workloads with available 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 threshold energy value at different value of resources. The workload is dispatched only, if the workload is executed within described budget and deadline and actual energy consumption is less than the threshold energy value. Dispatcher is used to dispatch the cloud workloads for execution. One cloud workload details and provided to the dispatcher.

A particular cloud workload is submitted to the compute resource for calculating the amount of resources required. The workload wrapper is created by allocating the resources to the cloud workloads. After that the tentative mapping is committed and each mapping is stored. The sequence diagram of dispatching workloads is shown in Fig. 12. The scaling listener is used to handle the request from various cloud consumers and allocate the cloud workloads to the available resources efficiently based on the workload details submitted by the cloud consumer. Resource secluding is done in two steps: in first step, the resource requirement for a particular cloud workload has been computed. In second step, discover processor properties like processing cost, speed and MIPS rating. After execution of every cloud workload, the processor properties are saved for future purpose. The resource scheduler schedules the incoming cloud workloads based on the workloads' details. First of all, get cloud workloads to schedule and then find appropriate and available resources and cloud workloads mapped efficiently based on the scheduling policies. All the incoming workloads are put into different queues than the queue that contains already submitted cloud workloads. The mapping is saved for future purpose.

Workload monitor is used to check the status of cloud workloads. The loop workload monitor is used to monitor the incoming cloud workloads. The queries related to

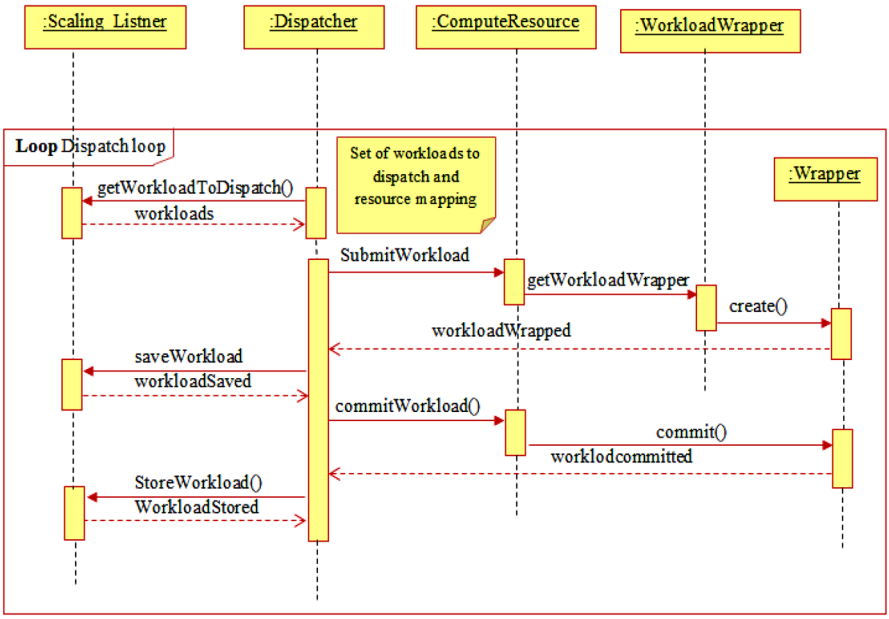

Fig. 12 Sequence diagram of dispatching workloads

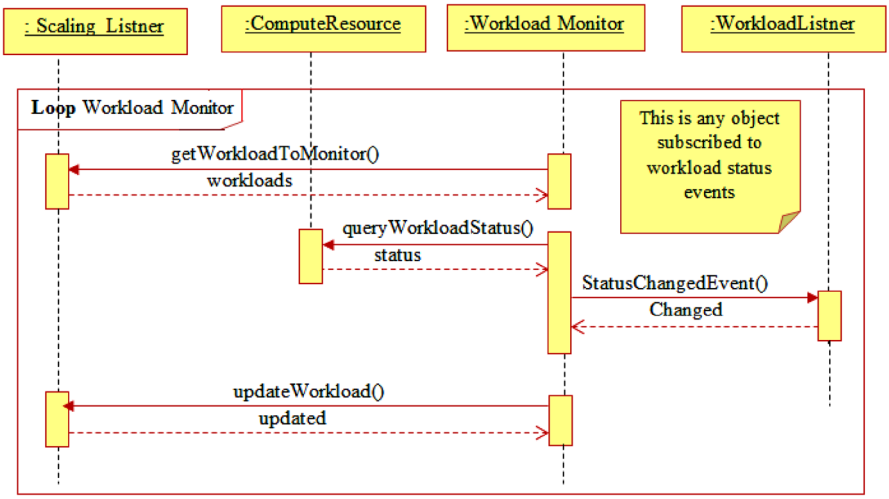

Fig. 13 Sequence diagram of workload monitoring

workload monitoring are asked and replied according to the status. After the execution of the cloud workload, the status is changed by workload listener from executing to finish. After the successful execution of cloud workload the status is updated in the system. The sequence diagram of workload monitoring is shown in Fig. 13. There are different clusters of cloud workloads created and put the incoming workloads in suitable cluster.

4 Experimental setup and results

Tools used for setting cloud environment are Microsoft Visual Studio 2010, NetBeans IDE 7.1.2, CloudSim 3.0, IntegratedNETJavaWeb and SQL Server 2008. Microsoft Visual Studio 2010 is an Integrated Development Environment from Microsoft. We have developed user interface of CWMF in .NET framework. The NetBeans IDE 7.1.2 is a modular, standards-based, Integrated Development Environment (IDE) written in the Java programming language. NetBeans is used to execute the CloudSim toolkit in which all the four scheduling policies have been implemented. The CloudSim toolkit supports both system and behavior modeling of cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies. Currently, it 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 provisioning techniques for allocation of VMs under inter-networked cloud computing scenarios [41]. The main advantages of using CloudSim for initial performance testing include: (1) time effectiveness: it requires very less effort and time to implement cloud-based application provisioning test environment and (2) flexibility and applicability: developers can model and test the performance of their application services in heterogenous cloud environments (Amazon EC2, Microsoft Azure) with little programming and deployment effort. CloudSim offers the following novel features:

- Support for modeling and simulation of large-scale cloud computing environments, including data centers, on a single physical computing node.
- A self-contained platform for modeling clouds, service brokers, provisioning, and allocation policies.
- Support for simulation of network connections among the simulated system elements.
- Availability of a virtualization engine that aids in the creation and management of multiple, independent, and co-hosted virtualized services on a data center node.
- Flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services.

The tool used to integrate Java with Microsoft Technology is called IntegratedNETJavaWeb. Through this tool some Java methods can call from .NET code, and pass some values to Java or .NET and vice versa. In this work, there is CWMP, i.e. ASP.NET application, which interacts with Java programs, i.e. CloudSim. The two IDEs for Application Visual Studio 2010 and NetBeans IDE 7.1.2 have been used. Microsoft SQL Server 2008 is a relational database management system developed by Microsoft. User information, workload details and resource details are stored in database through SQL Server.

4.1 Framework validation

Cloud workload management framework has been developed to optimize the workload schedule based on scheduling policies in simulated cloud environment. CWMF is used

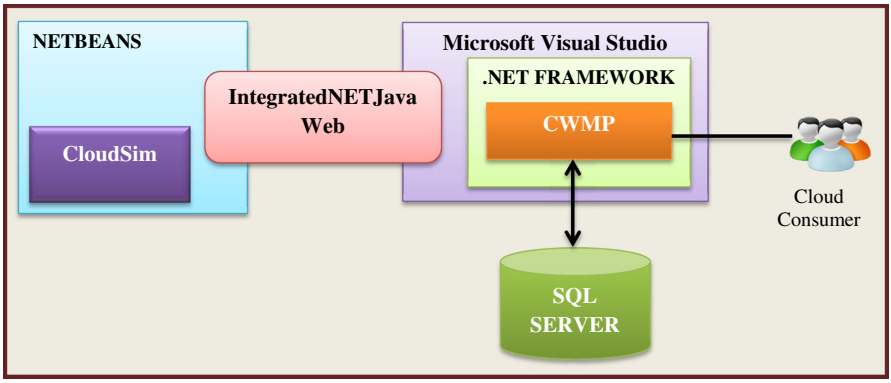

Fig. 14 Simulation environment

to execute the cloud workloads using available resources within the available budget and desired deadline. The mapping of resources to the cloud workloads is based on the scheduling policy. To continue with this system cloud consumer selects the workload name, desired deadline and preferred policy from the drop down list and enters the estimated budget (minimum $50). Based on workload detail, workload management system (WMS) will assign the workload type (compute, communication, administrator and storage) to every workload. Based on user details, the details of generated cloud workload schedule is entered to send these details to specific user. The workload schedule generated by cloud provider is displayed in the specific user account.

4.2 Simulation environment

The integration of multiple environments used to conduct experiments is shown in Fig. 14. Cloud user interacts with cloud workload management framework through CWMP to submit the workload details. User information, workload detail and resource detail are stored in database through SQL Server. Cloud workload management portal is implemented in .NET framework and framework is running in Microsoft Visual Studio. NetBeans is used to execute the CloudSim toolkit in which all the four 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.

4.3 Comparison with other simulators

Java-based simulator [40] and SwinDeW-C System [39] can execute only homogenous type of workflows. Java-based simulator [40] considers randomly generated workflows without maintaining any queue which leads to conflict in execution and wastage of time while SwinDeW-C System [39] not considered the communication time and communication cost during mapping of resources and workloads. But our simulation environment can be used to execute both homogenous and heterogeneous cloud workloads and considers both communication time and communication cost.

Table 13 Scheduling parameters and their values

Parameter	Value
Number of resources	50–250
Number of cloudlets (workloads)	3,000
Bandwidth	1,000–3,000 B/S
Size of cloud workload	10,000+ (10–30 %) MB
Number of PEs per machine	1
PE ratings	100–4,000 MIPS
Cost per cloud workload	$3–$5
Memory size	2,048–12,576 MB
File size	300 + (15–40 %) MB
Cloud workload output size	300 + (15–50 %) MB

4.4 Experimental results

3,000 independent cloud workloads were generated randomly in CloudSim as cloudlets. For each cloud workload, the attributes of the cloud workload include deadline, estimated budget and scheduling policy. In this paper, the six scenarios of the QoS distribution have been defined. For each set of resources, the attributes of the resource include number of resources (or computing nodes), QoS provided, and the information of each resources, which includes the deadline, estimated budget and scheduling policy. All configurations about the resources will remain the same during the experiment. In this paper, one-dimensional QoS (processing speed) has been implemented. The allocation agent receives the cloud workloads and puts them into the workload queue. While the workload queue is not empty, the allocation starts the scheduling policy to find the right workload-resource match according to the policy. To compare proposed QoS guided policies with the existing scheduling policies, all the proposed policies have been implemented to get the performance data. Machine Learning techniques are being used for making decisions based on some specified rules. Here, ten different cloud workloads along with identification number (WId) are considered to show how proposed (four) scheduling policies [CCTB scheduling policy, time-based (TB) scheduling policy, cost-based (CB) scheduling policy and bargaining-based (BB) scheduling policy], working in different criteria, perform better than already existing policies. The attributes of the cloud workload include deadline, estimated budget and policy.

To evaluate the effectiveness of the scheduling policies discussed in Sect. 3.5.5, the simulator namely CloudSim [41] has been used to calculate the execution time for each of them. In this section, the performance of all the algorithms is discussed. Table 13 shows the characteristics of resources and cloudlets that have been used for all the experiments. User cloud workloads are modeled as independent parallel applications are modeled which is 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.

Table 14 Cost related to different processing speeds

Service	PS (MIPS)	MIPS rating	Cost ($)/workload
Service 1	1	100	0
Service 2	2	250	50
Service 3	4	500	100
Service 4	8	1,000	200
Service 5	16	2,000	400
Service 6	32	4,000	800

Table 15 Deadline urgency

D_u	C_c ($)	C_t (s)
$D_u > 0.75$	1	$10/60 = 0.17$
$0.25 \leq D_u \leq 0.75$	3	$60/60 = 1$
$D_u < 0.25$	5	$120/60 = 2$

All the policies consider only one-dimensional QoS (processing speed) described in Table 14. The QoS is generated to be 1–32 with the ratio following given distribution of the QoS request.

The execution time (E_t) is calculated with the formula given below (Eq. 13):

$$E_t = \frac{D_t}{B_E \times \text{PS}} \times 100 \tag{13}$$

where D_t is deadline time, B_E is estimated budget and PS is processing speed.

4.4.1 Compromised cost-time based (CCTB) scheduling policy

Deadline urgency (D_u): it specifies cloud customer urgency to get workload (s) completed as specified in Eq. 14. This metric shows how the scheduler deals with cloud customer with different requirements on time. The value of communication cost (C_c) and communication time (C_t) depends on the deadline urgency (D_u) as shown in Table 15. Assign minimum cost to every workload based on user requirements.

$$D_u = \frac{D_t}{E_t} - 1 \tag{14}$$

The total expected cost (TEC) and total expected completion time (TECT) are calculated using the formula (Eq. 15) discussed in CCTB scheduling policy and the value for each workload will be calculated as shown in Table 16. For execution of cloud workload under this policy the condition will be fulfilled strictly as specified in Eq. 15:

$$\text{TECT} \leq D_t \;\&\&\; \text{TEC} \leq B_E \tag{15}$$

$$\text{Time difference } (T_d) = D_t - \text{TECT} \tag{16}$$

Table 16 Total expected cost and total expected completion time

WId	W_d (s)	B_E ($)	D_t (s)	E_t (s)	C_t (s)	C_C ($)	D_u	$C_{\min}$ ($)	TEC (s)	TECT (s)
W1	12:00	100	12	12	2	5	0	88	93	14
W2	4:00	62	4	6.45	2	5	−0.37	50	55	8.45
W3	6:00	120	6	5	2	5	0.2	112	117	7
W4	21:00	170	21	12.35	1	3	0.7	161	164	13.35
W5	10:00	155	10	6.45	1	3	0.55	152	155	8.45
W6	2:00	200	2	1	0.17	1	1	197	198	1.17
W7	4:00	252	4	1.58	0.17	1	1.53	246	247	1.75
W8	20:00	265	20	7.54	0.17	1	1.65	262	263	7.71
W9	4:00	72	4	5.55	2	5	−0.27	61	66	7.55
W10	14:00	65	14	21.53	2	5	−0.34	53	58	23.53

Table 17 Time difference (T_d)

WId	B_E ($)	D_t (s)	TEC	TECT	T_d	Deadline fulfilled
W1	100	12	105	14	−2	No
W2	60	4	65	8.45	−4.45	No
W3	120	6	125	7	−1	No
W4	170	21	173	13.35	7.65	Yes
W5	150	10	153	8.45	1.55	Yes
W6	200	2	201	1.17	0.83	Yes
W7	250	4	253	1.75	2.25	Yes
W8	260	20	263	7.71	12.29	Yes
W9	70	4	75	7.55	−3.55	No
W10	65	14	70	23.53	−9.53	No

where W_d is desired deadline in seconds, B_E is estimated budget in dollars, D_t is deadline time in seconds, C_t is communication time, E_t is execution time, $C_c$is communication cost, TECT is total expected completion time, D_u is deadline urgency and TEC is total expected cost. The time difference (T_d) is calculated by the formula defined above (Eq. 16) as shown in Table 17. If the value of $T_d > 0$ then the cloud workload will be executed before their deadline otherwise it will not be executed before deadline.

Number of deadlines missed: 5

4.4.1.1 Rescheduling of cloud workloads The workloads W4, W5, W6, W7 and W8 have extra time than required for execution within their deadline, the total extra time $(7.65 + 1.55 + 0.83 + 2.25 + 12.29) = 24.75$ Seconds and the time required to execute the pending workloads $(-2 - 4.45 - 1 - 3.55 - 9.53) = -20.53$ s is less than the time available. So the W1, W2, W3, W9 and W10 will be executed within their deadline and budget respectively as shown in Table 18.

Table 18 Rescheduling of cloud workloads

WId	D_t (H)	TECT	Updated D_t (s)	Deadline fulfilled
W1	12	14	14	Yes
W2	4	8.45	8.45	Yes
W3	6	7	7	Yes
W4	21	13.35	13.35	Yes
W5	10	8.45	8.45	Yes
W6	2	1.17	1.17	Yes
W7	4	1.75	1.75	Yes
W8	20	7.71	7.71	Yes
W9	4	7.55	7.55	Yes
W10	14	23.53	23.53	Yes

After rescheduling, deadline missed = 0

The implementation of compromised cost-time based scheduling policy follows the algorithm shown in Fig. 8. Calculate the TEC, TECT and time difference (T_d) to allocate the resources. The allocation agent finds the missed deadlines and calculates time difference for each workload then uses the extra available time to the workloads with missed deadlines and executes all the cloud workloads within their corresponding deadlines. The performance evaluation criterion to evaluate the performance of CCTB for resource scheduling has been defined. The two parameters namely cost and execution time has been selected. The cost and execution time measured in dollars and seconds, respectively. To validate CCTB scheduling policy, 3,000 cloud workloads and 50–250 resources are considered. To evaluate the performance of CCTB, the effects of different number of cloud workloads have been investigated. The two existing reference algorithms namely CTC [39] and DBD-CTO [40] have been used.

- Compromised time cost (CTC) scheduling algorithm—compromised cost-time scheduling policy considers the characteristics of cloud computing to accommodate instance-intensive cost-constrained workflows by compromising execution time and cost. The simulation performed demonstrates that the algorithm can cut down the mean execution cost and shorten the mean execution time within the user-designated execution cost [39].
- Deadline and budget distribution-based cost-time optimization (DBD-CTO) workflow scheduling algorithm that minimizes execution cost while meeting timeframe for delivering results and analyze the behavior of the algorithm. In this algorithm, the two constraints are considered: deadline and budget. For the workflow, a list of three services for each task of the workflow was created. The scheduler that is implemented in the broker part calls DBD-CTO to choose a particular service such that overall workflow execution should be in deadline and budget constraints specified by the user [40].

Both the algorithms are doing resource scheduling based on the homogenous cloud workloads without considering heterogeneous workloads. The concept of reschedul-

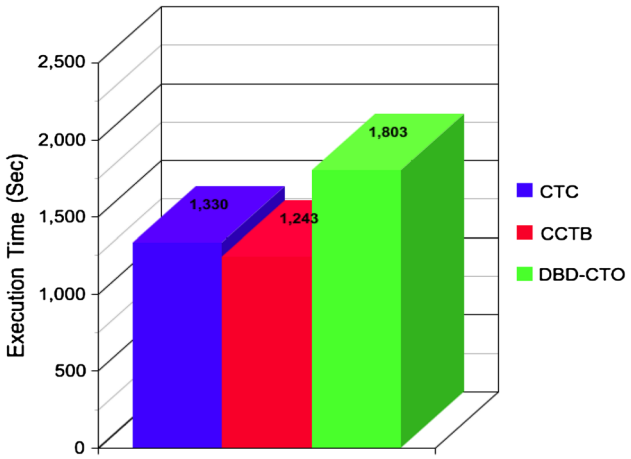

Fig. 15 Execution time comparison of cloud workloads

ing is also not used in these existed algorithms. All the three algorithms compromised time cost (CTC) scheduling algorithm (existing), deadline and budget distribution-based cost-time optimization (DBD-CTO) (existing) and CCTB (proposed) have been implemented in CloudSim by making the changes in the VMScheduler.java according to CCTB scheduling policy and then compared in different scenarios.

Test case 1: execution time comparison of cloud workloads: After executing the values on scheduling parameters for different algorithms, the cloud simulator CloudSim provides the execution summary for DBD-CTO, CTC and CCTB. The execution time for executing the same cloud workloads using same resources is 1,803 s in DBD-CTO whereas time taken to execute the same Cloud workloads by CTC algorithm is 1,330 s. CCTB is implemented and executed with same environment. The same cloud workload is executed in CCTB scheduling policy in 1,243 s. The execution time taken by CCTB scheduling policy is lesser than other scheduling algorithms. Figure 15 demonstrates the effectiveness of the CCTB scheduling policy in managing the time requirement of the cloud user.

The characteristics of cloudlets are used to compare the execution time of three algorithms. In this case lowest execution time was achieved in case of CCTB scheduling policy whereas DBD-CTO resulted in the highest execution time.

Test case 2: cost comparison of cloud workloads: Figure 16 shows the effect on cost by three algorithms. The cost for executing the same cloud workload using same resources is $340 in case of DBD-CTO whereas cost spent to execute the same workload by CTC is $220. The same workload is executed in CCTB scheduling policy is $170. Figure 16 shows the lowest cost achieved by CCTB where DBD-CTO resulted in highest cost.

Test case 3: execution time for different number of resources: Figure 17 shows the effect of increasing the number of resources, while keeping the number of cloud workloads being submitted to CWMF constant. In this experiment, 3,000 cloud workloads were executed with varying numbers of resources. The results depict that by increasing the

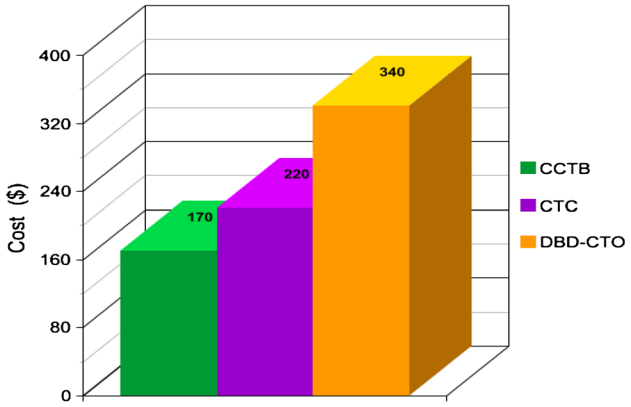

Fig. 16 Cost comparison of cloud workloads

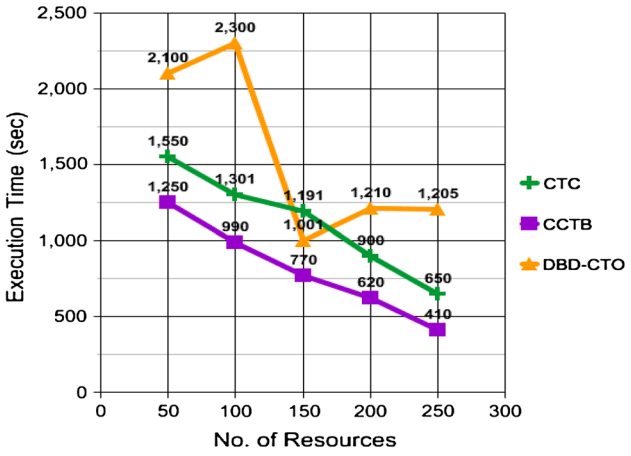

Fig. 17 Execution time for different number of resources

number of resources, the execution time decreases. CCTB scheduling policy performs better than DBD-CTO and CTC. Figure 17 shows the execution time decreases for CCTB and CTC in same proportion as we increase the number of resources.

This observation indicates that CCTB gives an equally good performance in comparison that given by CCTB and DBD-CTO.

Test case 4: cost for different number of resources: The cost of execution of different cloud workloads for three algorithms varies. The costs of resources are decreasing with increasing the number of resources. Figure 18 shows the CCTB scheduling policy executes the same number of cloud workloads at a minimum cost. The cost of workload execution is less using CCTB in comparison to the execution cost using CCTB and DBD-CTO. As the cost with Cloud resource is significant so the cost benefit (4–38 %) was notified with different number of resources. However, more benefit will be anticipated if the variations are higher.

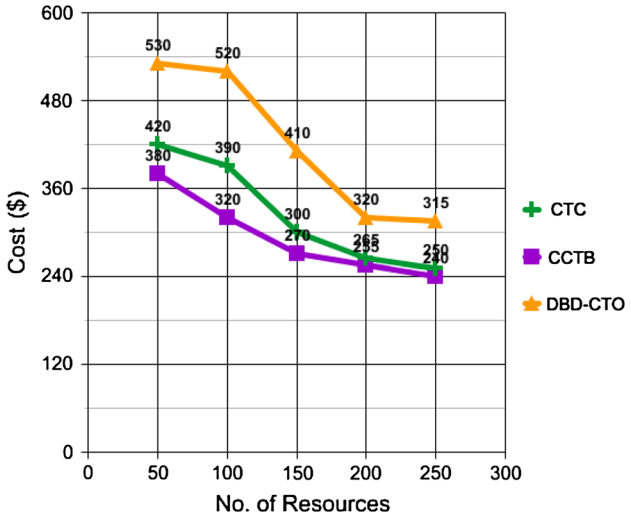

Fig. 18 Cost for different number of resources

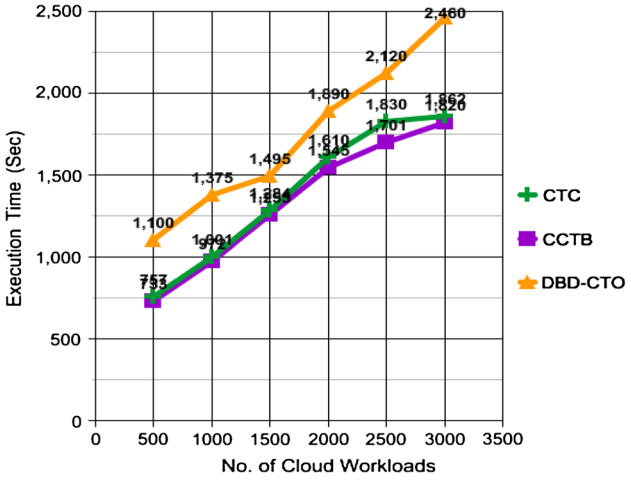

Fig. 19 Execution time for different number of cloud workloads

Test case 5: execution time for different number of cloud workloads: We have also performed experiments to determine the effect of an increase in number of workloads on cost and execution time. We have sent 3,000 workloads to cloud from the experiment result shown in Fig. 19, we can conclude that the time taken to execute workloads reduced using CCTB scheduling policy. The execution time reduction in execution is about 14–26 %. The execution time is increasing with the increase in number of cloud workloads and the execution time of CCTB for same number of cloud workloads is slightly lesser than CTC as shown in Fig. 19.

Test case 6: cost for different number of cloud workloads: Figure 20 shows that cost per cloud workload increases as the number of submitted cloud workload

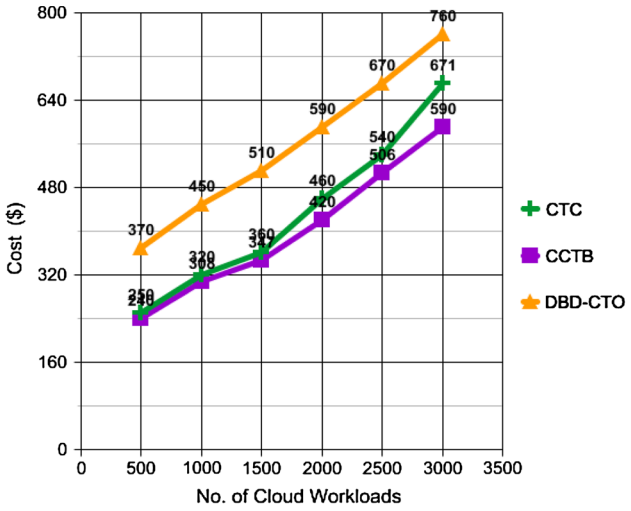

Fig. 20 Cost for different number of cloud workloads

increases. The existing algorithm based workload's execution resulted in a schedule which is expensive in comparison to the CCTB scheduling policy. From all the experimental results, the workload execution using the CCTB scheduling policy performs better. The overall cost for cloud consumer's workload execution is less.

Our scheduling policy performed better with more number of cloud workloads as compared to existing scheduling policies.

Test case 7: number of missed deadlines: There are different numbers of deadlines missed in different algorithms. With increasing the number of cloud workloads, the number of deadlines missed is also increasing. The number of deadlines missed in Deadline and budget distribution-based cost-time optimization (DBD-CTO) is maximum and minimum in CCTB scheduling policy as shown in Fig. 21.

The variation in number of deadlines missed at 500 workloads is lesser as compared to the 3,000 workloads.

Test case 8: execution time variation: The execution time is decreasing with the increase in budget as shown in Fig. 22. In this research work, minimum cost for execution is $50. At a minimum budget, the execution time is larger. With the increase in budget, the more number of resources provided to reduce the execution time and number of deadlines missed are also decreasing. There is slight reduction in execution time with budget ($150–$200).

Test case 9: execution cost variation: The cost of executing the cloud workloads is varying with the increase in allocated budget as shown in Fig. 23. Execution cost is increasing from minimum budget to maximum budget in same proportion approximately.

Test case 10: energy consumption: For different number of resources there is fixed threshold value of energy consumed during cloud workload execution. The calculated energy consumption is compared with threshold value and the cloud workload is exe-

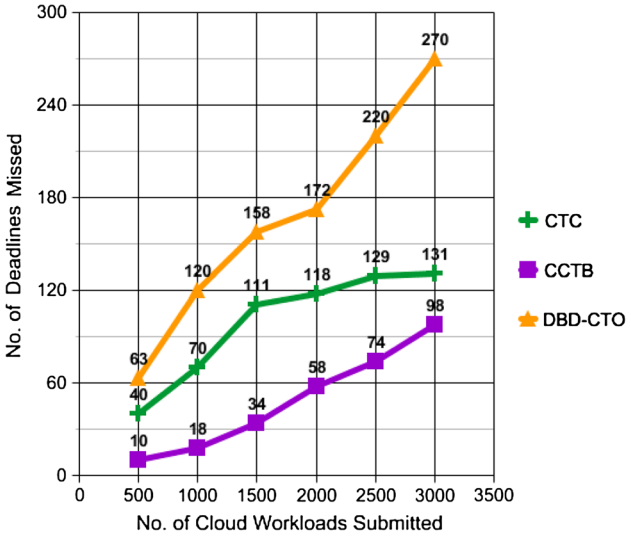

Fig. 21 Number of missed deadlines

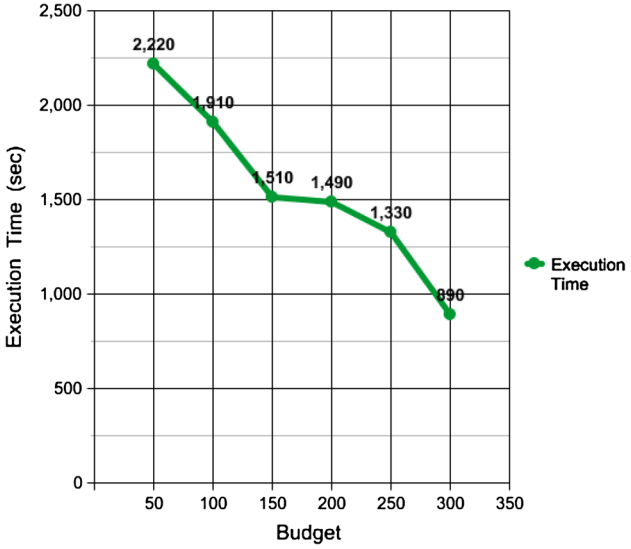

Fig. 22 Execution time variation

cuted only if the calculated energy consumption is less than or equal to threshold value. For successful execution of a cloud workload, the actual energy consumption (E_{Cloud}) will be less than threshold energy value ($E_{\text{Threshold}}$). Figure 24 shows the comparison of actual energy consumption and threshold energy value. Energy consumption is reduced with increase in number of resources. The actual energy consumption (E_{Cloud}) reduces up to 9.99 % at 100 resources and 11.02 % at 300 resources.

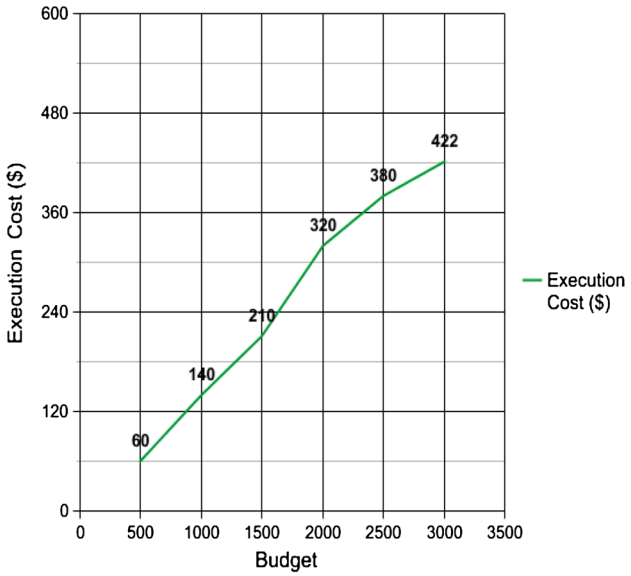

Fig. 23 Execution cost variation

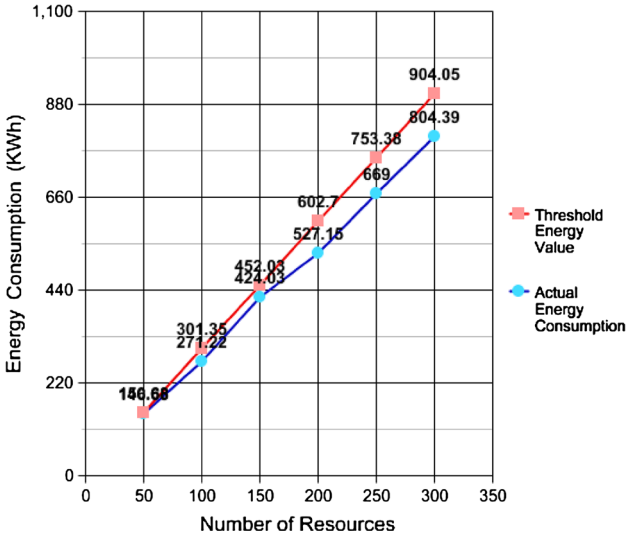

Fig. 24 Energy consumption and number of resources

4.4.2 Cost-based (CB) scheduling policy

Allocate resources based on cost, the workload which has more budgets (B_E) will execute first. If the two workloads have same budget then that workload will execute first that has lesser execution time. By default, PS = 1.

The implementation of cost-based scheduling policy follows the algorithm shown in Fig. 9. First, the allocation agent begins to compute the cost of each cloud workload

B_E	265	252	200	170	155	120	100	72	65	62
Workload	W8	W7	W6	W4	W5	W3	W1	W9	W10	W2
Resource	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10

then sorted as the priority given to the cloud workload which has maximum budget. 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.

4.4.3 Time-based (TB) scheduling policy

Allocate resources based on time, the workload which has shortest deadline time (D_t) will execute first. If the two workloads have same deadline time then that workload will execute first that has lesser execution time. By default, PS = 1. In this scenario, there are three workloads (W2, W7, W9) having same deadline time, sorted according to the maximum budget (W7 > W9 > W2).

D_t	2	4	4	4	6	10	12	14	20	21
Workload	W6	W7	W9	W2	W3	W5	W1	W10	W8	W4
Resource	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10

Number of deadlines missed: 3 (W2, W9, W10)

To execute W2, W9 and W10 within their deadline, change budget according to the processing speed, and add cost $50 if user wants to double the processing speed (see Table 14). With PS = 2, these three workloads will be executed before deadline, by recalculating the value of E_t as described in Table 19.

Now the workloads W2, W9 and W10 will be executed before their deadline.

The implementation of Time-Based Scheduling Policy follows the algorithm shown in Fig. 10. First, the allocation agent begins to compute the deadline time of the cloud workload on the given budget. The allocation agent then schedules all the cloud workloads with smallest execution time request to the resources that provide high QoS. If any deadline found missed then recalculate the execution time by increasing the value of processing speed (PS) and it will increase cost only.

Test case 1: completion time vs. allocated budget: The cloud workload is being executed with different constraints. There is a fixed deadline, i.e. 1,500 s; the cloud work-

Table 19 Actual and improved execution time

WId	D_t (s)	Actual E_t (PS = 1)	Improved E_t (PS = 2)
W2	4	6.45	3.22
W9	4	5.55	2.775
W10	14	21.53	10.765

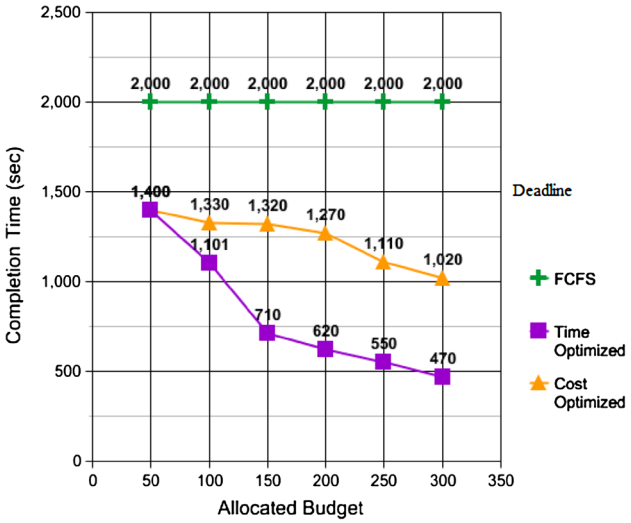

Fig. 25 Completion time vs. allocated budget

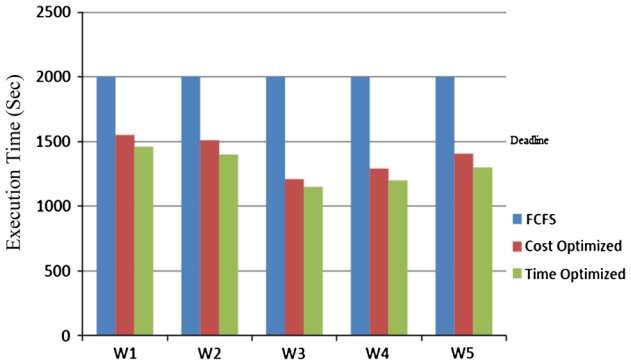

Fig. 26 Completion time of different workloads

load should be executed successfully before deadline. In this experiment, different budgets were allocated. In first come first serve (FCFS) based scheduling policy, the execution time is larger at different budgets. The execution time in cost-based scheduling policy is lesser than FCFS but more than time-based scheduling policy. The execution time is lesser in time-based scheduling policy at maximum budget that has been allocated. The execution time in both the cost and time-based scheduling policy is decreasing but remains same in FCFS as shown in Fig. 25. The completion time is reduced to 53.9 % at the maximum budget, i.e. $300.

Test case 2: completion time of different workloads: The execution time required to complete the cloud workload (W) is maximum in FCFS and minimum in cost-based scheduling policy, but least in time-based scheduling policy as shown in Fig. 26. Both cost and time-based scheduling policy executes the cloud workload before desired deadline.

Table 20 Cloud workloads detail

WId	W_d (s)	B_E ($)	E_t
W1	12:00	100	12
W2	4:00	62	6.45
W3	6:00	120	5
W4	21:00	170	12.35
W5	10:00	155	6.45
W6	2:00	200	1
W7	4:00	252	1.58
W8	20:00	265	7.54
W9	4:00	72	5.55
W10	14:00	65	21.53

Table 21 Resource detail

Time slot (s)									
0–2		2–4		4–8		8–16		16–32	
R	C ($)	R	C ($)	R	C ($)	R	C ($)	R	C ($)
R1	100	R5	80	R8	105	R12	80	R16	160
R2	90	R6	75	R9	115	R13	85	R17	180
R3	110	R7	110	R10	125	R14	90	–	–
R4	120	–	–	R11	90	R15	70	–	–

4.4.4 Bargaining-based (BB) scheduling policy

Table 20 shows different cloud workloads along with their parameters such as deadline, estimated budget and execution time.

The various resources (R) available for the execution of above cloud workloads along with their execution cost (C) and classified according to time slots (seconds) are described in Table 21.

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 as shown below:

E_t	12	6.45	5	12.35	6.45	1	1.58	7.54	5.55	21.53
Workload	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
Resource	R15	R11	R11	R15	R11	R2	R2	R11	R11	R16

The implementation of bargaining-based scheduling policy (BBSP) follows the algorithm shown in Fig. 11. The implementation complies with the negotiation among the various resources and cloud workload producer along with different time slots. The

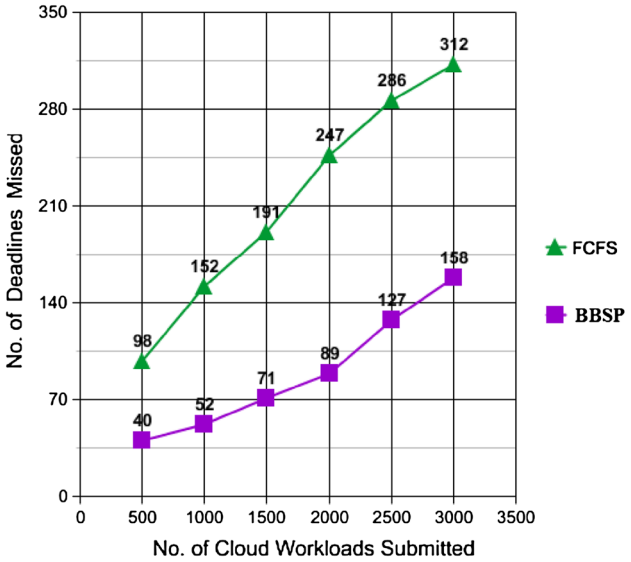

Fig. 27 Number of deadlines missed

allocation agent allocates the resources based on the bargaining between them. The number of missed deadlines in both the cases is shown in Fig. 27.

With the increase in number of cloud workloads, the rate of missed deadlines is increased. In both the policies, the number of deadlines missed is varying. The lesser number of deadlines missed in BBSP as compared to FCFS.

4.5 Statistical analysis

Statistical significance of the results has been analyzed by coefficient of variation (CoV), a statistical method. CoV is statistical measure of the distribution of data about the mean value. CoV is used to compare to different means and furthermore offers an overall analysis of performance of the technique used for creating the statistics. It states the deviation of the data as a proportion of its average value, and is calculated as follows (Eq. 17):

$$\text{CoV} = \frac{\sigma}{\mu} \times 100 \tag{17}$$

where σ is a standard deviation and μ is mean. CoV of execution time and cost has been studied of Cloud workload of every scheduling policy as shown in Figs. 28 and 29.

CoV calculated for execution time and cost results attained by CTC, CCTB and DBD-CTO resource scheduling polices. Range of CoV (0.7–3 %) for execution time and (0.5–2 %) for cost approves the stability CCTB resource scheduling policy as shown in Figs. 28 and 29.

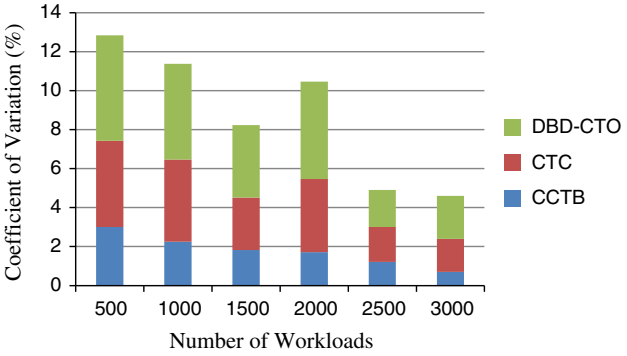

Fig. 28 CoV for execution time with each secluding policy

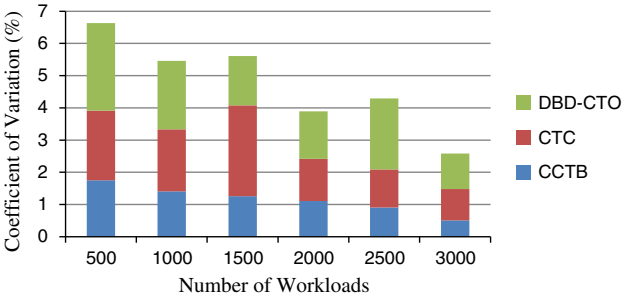

Fig. 29 CoV for cost with each secluding policy

Small value of CoV signifies CCTB is more efficient in resource scheduling in the situations where the number of cloud workloads has changed. Value of CoV decreases as the number of workloads is increasing. Statistical analysis demonstrates the CCTB outperforms other resource scheduling polices for large numbers of cloud workloads. With small value of CoV system is more stable and CCTB resource scheduling policy attained the best results in the cloud for cost and execution time as QoS parameters.

4.6 Discussions

The performance of resource scheduling policies of CWMF has been compared with existing resource scheduling polices. The performance of CWMF has been analyzed with different number of cloud workloads and resources. The performance of framework has been evaluated with respect to execution time, cost and energy consumption. Execution cost permits the evaluation for selection of resource whereas duration of workload execution evaluates by execution time. Dynamic voltage supply can be adjusted by energy consumption. The policy in which cloud provider minimizes the cost as well as execution time along with least energy consumption is called CCTB scheduling policy. CCTB reduces the execution time by up to 30.94 % compared to deadline and budget distribution-based cost time optimization (DBD-CTO) and 6.54 %

for CTC. CCTB reduces the execution cost by up to 50 % compared to DBD-CTO and 22.72 % for compromised time cost (CTC). The results depict that by increasing the number of resources, the execution time decreases. The costs of resources are decreasing with increasing the number of resources. The CCTB scheduling policy executes the same number of cloud workloads at a minimum cost. The execution time is increasing with the increase in number of cloud workloads and the execution time of CCTB for same number of cloud workloads is slightly lesser than CTC. The existing algorithm based workload's execution resulted in a schedule which is expensive in comparison to the CCTB scheduling policy. From all the experimental results, the workload execution using the CCTB scheduling policy performs better. The overall cost for cloud consumer's workload execution is less. The number of deadlines missed in DBD-CTO is maximum and minimum in CCTB. With the increase in budget, the more number of resources provided to reduce the execution time and number of deadlines missed are also decreasing. The cost of executing the cloud workloads is varying with the increase in allocated budget. The actual energy consumption (E_{Cloud}) reduces up to 9.99 % at 100 resources and 11.02 % at 300 resources. Considering all these QoS parameters (execution time, cost and energy consumption) and simulation results, it is found that the CWMF provides a better solution for heterogenous cloud workloads and approximates optimal solution for resource scheduling challenges. Approximate cost to implement the proposed framework in real cloud environment depends upon the number of resources required to fulfill the current demand and QoS requirements submitted by the cloud consumer.

5 Conclusions

Cloud computing and its vital characteristics have been discussed in this paper. This paper focuses on the resource scheduling challenges that cloud computing is facing today. Several resource scheduling algorithms are compared with respect to the cloud workload as an issue for the dynamic scalability of resources. This paper also gives an insight about the problem of making decisions based on cost and time constraints in cloud computing. The different cloud workloads have been identified and analyzed. The QoS requirements for every cloud workload have been identified. The clustering of these cloud workloads is done through K-means clustering algorithm by assigning the appropriate weights to the different quality attributes. Further, workload based cloud framework is proposed and implemented in this work. The experimental results gathered through CloudSim clearly demonstrate that the proposed framework has better performance in terms of execution time, cost and energy consumption as compared to existing scheduling algorithms.

6 Future directions

Our proposed framework maps and executes the workloads based on workloads' details given by user and resource details given by providers. In the future, we will further develop an autonomic resource scheduling technique that efficiently schedules the provisioned cloud resources and maintains the SLA based on user's QoS

requirements to reduce the above mentioned dependency. The proposed framework presented in this paper can be extended further to add sensitivity of assumptions in weight calculations of both homogenous and heterogenous cloud workloads. IaaS providers can use these results to quickly assess possible reductions in execution time and cost, hence having the potential to save energy. CloudSim toolkit has been used to collect the current results but the same results would be verified on actual Cloud resource present at Center of Excellence (CoE) in Grid Computing at Thapar University.

Acknowledgments One of the authors, Sukhpal Singh, gratefully 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 like to thank Dr. Maninder Singh for helping in improving the language and expression of preliminary version of paper.

References

1. Armando MF, Rean G, Anthony DJ, Randy K, Andy K, Gunho L, David P, Ariel R, Ion S, Matei Z (2010) A view of cloud computing. Commun ACM 53(4):50–58
2. Rimal BP, Jukan A, Katsaros D, Goeleven Y (2011) Architectural requirements for cloud computing systems: an enterprise cloud approach. J Grid Comput 9(1):3–26
3. Rimal P, Choi E (2009) A taxonomy and survey of cloud computing systems. In: Fifth international joint conference on INC, IMS and IDC, Seoul, Korea
4. Buyya R, Yeoa CS, Venugopala S, Broberga J, Brandicc I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616
5. Dillon T, Wu C, Chang E (2010) Cloud computing: issues and challenges. In: 24th IEEE international conference on advanced information networking and applications, Perth, Australia
6. Jesús JD (2013) Cloud deployment models, IBM, (Online). http://www.ibm.com/developerworks/websphere/techjournal/1206_dejesus/1206_dejesus.html. Accessed 14 Jan 2013
7. Singh S, Chana I (2012) Cloud based development issues: a methodical analysis. Int J Cloud Comput Serv Sci 2(1):73–84
8. Cirne W, Berman F (2001) A comprehensive model of the supercomputer workload. In: Fourth annual IEEE international workshop on workload characterization, WWC-4, Austin, Texas
9. Gmach D, Rolia J, Cherkasova L, Kemper A (2007) Workload analysis and demand prediction of enterprise data center applications. In: IEEE 10th international symposium on workload characterization, IISWC '07, Boston, MA, USA
10. Cherkasova L, Gupta M (2002) Characterizing locality, evolution, and life span of accesses in enterprise media server workloads. In: 12th international workshop on Network and operating systems support for digital audio and video, FL, USA
11. Rolia J, Cherkasova L, Arlitt M, Andrzejak A (2005) A capacity management service for resource pools. In: 5th international workshop on software and performance, Illes Balears, Spain
12. Arlitt MF, Williamson CL (1996) Web server workload characterization: the search for invariants. In: ACM SIGMETRICS international conference on measurement and modeling of computer systems, SIGMETRICS '96, Philadelphia, PA, USA
13. Bobroff N, Kochut A, Beaty K (2007) Dynamic placement of virtual machines for managing SLA violations. In: IFIP/IEEE integrated network management, Bombay
14. Verma A, Dasgupta G, Kumar T, Prad N (2009) Server workload analysis for power minimization using consolidation. In: USENIX annual technical conference, San Jose, CA
15. Khan A, Yan X, Tao S, Anerousis N (2012) Workload characterization and prediction in the cloud: a multiple time series approach. In: Network operations and management symposium (NOMS), IEEE, Krakow, Poland

16. Chen SJ, Liang PH, Yang J-M (2010) Workload evaluation and analysis on virtual systems. In: IEEE international conference on E-business engineering, 2010
17. Bossche RVD, Vanmechelen K, Broeckhove J (2010) Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In: IEEE 3rd international conference on cloud computing, Florida, USA
18. Xiong P, Wang Z, Malkowski S, Wang Q, Jayasinghe D, Pu C (2011) Economical and robust provisioning of n-tier cloud workloads: a multi-level control approach. In: Distributed computing systems (ICDCS), Minneapolis, Minnesota
19. Tsakalozos K, Roussopoulos M, Floros V, Delis A (2010) Nefeli: Hint-based execution of workloads in clouds. In: International conference on distributed computing systems, Genova, Italy
20. Bossche RVD, Vanmechelen K, Broeckhove J (2013) Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. Future Gener Comput Syst 29(4):973–985
21. Kousiouris G, Cucinotta T, Varvarigou T (2011) The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks. J Syst Softw 84(8):1270–1291
22. Mahambre S, Kulkarni P, Bellur U, Chafle G, Deshpande D (2012) Workload characterization for capacity planning and performance management in IaaS cloud. In: IEEE cloud computing in emerging markets (CCEM), Bangalore, India
23. 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
24. Topcuoglu H, Hariri S, Wu M-Y (1999) Task scheduling algorithms for heterogeneous processors. In: Heterogeneous computing workshop, (HCW'99), San Juan, Puerto Rico
25. El-kenawy E-ST, El-Desoky AI, Al-rahamawy MF (2012) Extended max–min scheduling using petri net and load balancing. Int J Soft Comput Eng (IJSCE) 2(4):198–203
26. Varalakshmi P, Ramaswamy A, Balasubramanian A, Vijaykumar P (2011) An optimal workflow based scheduling and resource allocation in cloud. In: Advances in computing and communications. Springer, Berlin, Heidelberg, pp 411–420
27. Li, Kun, Gaochao Xu, Guangyu Zhao, Yushuang Dong, and Dan Wang. "Cloud task scheduling based on load balancing ant colony optimization". In Chinagrid Conference (ChinaGrid), 2011 Sixth Annual, pp. 3–9. IEEE, 2011.
28. Xu M, Cui L, Wang H, Bi Y (2009) A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing. In: IEEE international symposium on parallel and distributed processing with applications, MA, USA
29. Ambike S, Bhansali D, Kshirsagar J, Bansiwal J (2012) An optimistic differentiated job scheduling system for cloud computing. Int J Eng Res Appl (IJERA) 2(2):1212–1214
30. Yu J, Buyya R, Tham CK (2005) Cost-based scheduling of scientific workflow applications on utility grids. In: E-Science and grid computing, IEEE, IL, USA
31. Sakellariou R, Zhao H, Tsiakkouri E, Dikaiakos MD (2007) Scheduling workflows with budget constraints. In: Integrated research in GRID Comput, pp 189–202
32. Selvarani S, Sadhasivam GS (2010) Improved cost-based algorithm for task scheduling in cloud computing. In: IEEE, computational intelligence and computing research (ICCIC), Tamilnadu, India
33. Dakshayini M, Guruprasad HS (2011) An optimal model for priority based service scheduling policy for cloud computing environment. Int J Comput Appl 32(9):0975–8887
34. S. Ghanbari and. M. Othman, "A Priority based Job Scheduling Algorithm in Cloud Computing", Procedia Engineering, International Conference on Advances Science and Contemporary Engineering, vol. 50, pp. 778–785, 2012.
35. Wu Z, Liu X, Ni Z, Yuan D, Yang Y (2013) A market-oriented hierarchical scheduling strategy in cloud workflow systems. J Supercomput 63(1):256–293
36. Delavar AG, Javanmard M, Shabestari MB, Talebi MK (2012) RSDC (reliable scheduling distributed in cloud computing). Int J Comput Sci Eng Appl (IJCSEA) 2(3):1–16
37. Moschakis IA, Karatza HD (2012) Evaluation of gang scheduling performance and cost in a cloud computing system. J Supercomput 59(2):975–992
38. Zhan Z-H, Zhang J, Li Y, Chung HS-H (2009) Adaptive particle swarm optimization. IEEE Trans Syst Man Cybern-Part B: Cybern 39(6):1362–1381

39. 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
40. Verma A, Kaushal S (2012) 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)
41. Calheiros RN, Ranjan R, Rose CAFD, Buyya R (2009) CloudSim: a novel framework for modeling and simulation of cloud computing infrastructures and services. In: Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia
42. Singh S, Chana I (2014) Energy based efficient resource scheduling: a step towards green computing. Int J Energy Inf Commun 5(2):35–52
43. Singh S, Chana I (2014) Metrics based workload analysis technique for IaaS cloud. In: Proceedings of international conference on next generation computing and communication technologies, 23rd and 24th April 2014, Dubai
44. Omer K, Maljevic I, Anthony R, Petridis M, Parrott K, Schulz M (2011) Dynamic scheduling of virtual machines running HPC workloads in scientific grids. In: 3rd international IEEE conference on new technologies, mobility and security (NTMS)
45. Cloud workloads (2013) CloudRoad, (online). http://www.1Cloudroad.com/Cloud-infrastructure-providers-for-2013. Accessed 11 Feb 2013
46. Qureshi MRJ, Qureshi WA (2012) Evaluating requirement specification document to improve the quality of software. AWERProc Inf Technol Comput Sci 1:596–600
47. Elghany MA, Khalifa N (2012) Quantifying software reliability attribute through the adoption of weighting approach to functional requirements. In: International conference on software and computer applications (ICSCA 2012), Singapore
48. Saeid M, Ghani AAA, Selamat H (2011) Rank-Order Weighting of Web Attributes for Website Evaluation. The International Arab Journal of Information Technology 8(1):30–38
49. Dromey RG (1995) A model for software product quality. IEEE Trans Softw Eng 21(2):146–462
50. Malik SU (2012) Customer satisfaction, perceived service quality and mediating role of perceived value. Int J Mark Stud 4(1):68–76
51. Nallur V, Bahsoon R (2010) Design of a market-based mechanism for quality attribute tradeoff of services in the cloud. In: ACM symposium on applied computing
52. Stefani A, Xenos M (2008) E-commerce system quality assessment using a model based on ISO 9126 and belief networks. Softw Qual Control 16(1):107–129
53. Davoudi M, Aliee FS (2009) A new AHP-based approach towards enterprise architecture quality attribute analysis. In: Research challenges in information science, RCIS
54. Garofalakis J, Stefani A, Stefanis V, Xenos M (2008) Quality attributes of consumer-based M-commerce systems, White Paper, University of Patras
55. Otieno C, Mwangi W, Kimani S (2012) Framework to assess software quality in ERP systems. In: Scientific conference proceedings
56. Clements P, Kazman R, Klein M (2002) Evaluating software architectures: methods and case studies. Addison-Wesley Longman, Boston, MA, USA
57. Meiappane A, Venkatesan VP, Roshini N, Nivedha S, Maheswar R (2013) Evaluation of software architecture quality attribute for an internet banking system. Int J Sci Eng Res 4(4):1704–1708
58. Stefani A, Xenos MN (2009) Meta-metric evaluation of E-commerce-related metrics. Electr Notes Theor Comput Sci (ENTCS) 233:59–72
59. Bhattacharjee PK (2010) Service quality measurement with minimum attributes (SERVQUAL-MA) technique upgrade by human resource development. Int J Innov Manage Technol 1(3):322–327
60. Barbacci MR (2003) Software Quality Attributes and Architecture Tradeoffs. Carnegie Mellon University, Pittsburgh, Software Engineering Institute
61. Berander P, Damm L-O, Eriksson J, Gorschek T, Henningssonv, Jönsson P, Kågström S, Milicic D, Mårtensson F, Rönkkö K, Tomaszewski P (2005) Software quality attributes and trade-offs. Blekinge Institute of Technology
62. Brooke J (1996) SUS-A quick and dirty usability scale. Redhatch Consulting, UK
63. Singh RV, Bhatia MP (2011) Data clustering with modified K-means algorithm. In: IEEE-international conference on recent trends in information technology, ICRTIT, Chennai
64. Gupta GK (2006) Introduction to data mining with case studies. PHI Learning Pvt. Ltd., New Delhi

65. Su M-C, Chou C-H (2001) A modified version of the k-means algorithm with a distance based on cluster symmetry. IEEE Trans Pattern Anal Mach Intell 23(6):674–680
66. Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY (2002) An efficient k-means clustering algorithm: analysis and implementation. IEEE Trans Pattern Anal Mach Intell 24(7):881–892
67. Mahendiran A, Saravanan N, Subramanian NV, Sairam N (2012) Implementation of K-means clustering in cloud computing environment. Res J Appl Sci Eng Technol 4(10):1391–1394
68. Barseghyan A, Kupriyanov M, Kholod I, Yelizarov S, Thess M (2014) Analysis of data and processes: from standard to realtime data mining. Re Di Roma-Verlag, Remscheid, Germany
69. V'azquez C, Huedo E, Montero RS, Llorente IM (2009) Dynamic provision of computing resources from grid infrastructures and cloud providers. In: Workshops at the grid and pervasive computing conference