Bhattacharya *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2019) 8:14
https://doi.org/10.1186/s13677-019-0137-8

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Replaceability and negotiation in a cloud service ecosystem

Adrija Bhattacharya[1*], Sankhayan Choudhury[1] and Agostino Cortesi[2]

Abstract

Cloud federation is an aggregation of services from different providers in a single pool supporting interoperability and resource migration. In federation, Services are assigned to the consumer's service access pool as per their specific functional and associated Quality level requirements. The said assignment is based on the advertised features of services. Sometimes, the selected provider fails to provide the committed service or, it fails to fulfill the expected QoS level. As a result, the consumer is being deprived of getting the services at required quality levels, in spite of subscribing and paying. Re-federation i.e. the inclusion of new services from different providers in the resource pool is a solution. This costly and time consuming re-federation process harms the overall harmony, reputation and performance of the existing federation. In this paper, the necessary strategies to make a federation autonomic is proposed. It helps federation to work in a self-adaptive manner by delaying the re-federation process through replacement and negotiation mechanisms. This allows the federation to keep a balanced state in case of failures. The proposed methods are simulated and the claims are substantiated by the preliminary experimental outcomes.

Keywords: Cloud federation, Ecosystem, QoS, Replacement, Broker negotiation

Introduction

In the competitive cloud market, Cloud Service Providers (CSPs) advertise their offerings and service consumers choose the most suitable one from the offerings according to their need. Consumer comes with typical requirements in terms of functionalities and associated Quality of Service parameters (QoSs). The said request can be satisfied by a composed service generated through the execution of service discovery and composition in sequence. A composed service is represented by a typical workflow of atomic services. The selected workflow is capable to render the functionalities, as requested, with desired QoS levels. But in a real time provisioning, the provider may fail to offer the committed services with the advertised QoS level. Moreover, due to some unavoidable situations, a provider may fail to render one (or more) of the constituent services of the selected workflow. As a result the consumer is not getting the subscribed services with desired QoS levels.

A cloud federation is constructed of multiple cloud providers that have easy collaborations. This could help to elevate the cloud service performances into more efficient one [8]. The collaborated service providers in a cloud federation contribute to construct a service pool containing all services of all providers. The service requirement for any consumer belongs to the federation is fulfilled by a set of candidate services from this service pool. The challenges in multi-cloud scenario can be solved by cloud brokers also. Broker is defined formally in [14]. It works as a middleman between the service provider(s) and consumer. Broker takes consumer's service request and tries to find out plausible solution from multiple clouds. Then it provides an integrated solution to the consumer through this brokerage.

Unfortunately, the provisioning in real time within a federation fails due to the challenges stated above. In general, there exist a set of services (after discovery) that can serve the request, but one of them can be consumed. The approach to restore the previous situation is to replace the failed service with the alternatives. This concept of replacement is available in the domain of web services with respect to workflow composition [2].

*Correspondence: adrija.bhattacharya@acm.org
[1]University of Calcutta, Kolkata, India
Full list of author information is available at the end of the article

A concept of replacement can be used for risk mitigation in federation. The provisioning and replacement of services within pool under federation could result an "out of resource" situation. In that case, newer service providers with demanding services are included in the pool. This is referred as a process of re-federation [1]. This is a costly and time consuming process. In this paper, we propose a mechanism that can effectively maximize the usage of an existing service resource pool within the federation and delays the re-federation process as much possible.

Re-federation can be triggered due to the following incidents

1. The CSP deviated from the agreed quality. The failure of QoS delivery cannot be met by existing available services in the resource pool.
2. New requests cannot be served by existing services in the resource pool.

The maintenance of the agreed service quality level is important throughout the period of service provisioning. There may be some breach of SLA (Service Level Agreement) if the quality parameter significantly deviated from the subscribed one. Here, 9 QoS parameters (Response time, Availability, Reliability, Throughput, successability, Latency, Compliance, Best Practices and Documentation) are considered for SaaS and standard QoSs are considered for IaaS. Our objective is to restore the previous state without performing a re-federation through a proposed alternative. The challenge of delaying re-federation can be resolved through the solution of the following issues.

1. How to maintain the service quality levels on the fly in case of failure of an atomic subscribed service?
2. How to accommodate a new service request (with subsequent QoS) in case of unavailability of resource(s)?
3. How to fulfill the increased QoS request for an ongoing allocated service by a consumer?

The rest of the paper is organized as follows: The scope of the problem is described in "Scope of the work" section and its formal definition is discussed in "Problem formulation" section. Service ecosystem definition and related terms are introduced in "Service ecosystem and related term definitions" section. "Solution proposal" section describes the proposed algorithm. Implementation and experimental results are depicted in "Experiments" section. "State of the art" section discusses the Related work. Finally, "Conclusion" section concludes.

State of the art

Cloud Federation refers to the unionization of software, infrastructure and platform services [10]. In federated Cloud one CSP helps to serve the requests of other CSPs [9]. Celesti et al. [8] has an approach of establishing federation by a software component in charge of executing the three main functionalities required for a federation. These three functionalities are identified as discovery of service information, matching the requirement and services and authenticating those services. In [6], Federation approach extends the idea of market-oriented side of clouds. In that approach the hint towards the cloud exchange policies were discussed as a future work. Further in [5], a federation with opportunistic and scalable application environment was created. Research issues were pointed out in [5] as flexible mapping, user specific QoS optimization, enterprise integration and scalable monitoring of system components. In another approach [13], a federated environment is conceptualized on top of EASI-Cloud Project. This mainly focuses on the infrastructure part; such as VM migration, scheduling, performance optimization etc. The challenges found in EASI-Cloud documentation with respect to SaaS offering is limited. Moreover, the main challenge (that is someway unachievable) in EASI-Cloud is with respect to SLA-monitoring. Another challenge as mentioned by the final reporting of EASI-Cloud was absence of offer (as provided by many providers) comparison. Comparison among the service prices and QoS monitoring was mainly unachieved. At another part in EASI-Cloud project, a tool for migrating application to SaaS was introduced.The challenge faced in this part was to make the service scalable and easy deployable on top of cloud's virtual infrastructure resources. However, in case of SLA violation, a proper alternative has to be found out to satisfy consumers. In most of the discussed approaches only have some simulated experiments that does not included any replacement strategies or SLA negotiation proposed in SaaS layer. Moreover, the QoS monitoring only in IaaS level restricts the actual application deployment and concerned QoS parameters (in SaaS layer) are deprived of actual performance assessment.

In current scenario, majority of the cloud federation approaches work in IaaS layer. However, it is clear from the above discussion that QoS consideration and monitoring for SaaS provisioning within cloud federation is a challenging area to contribute. Let us discuss some parallel approaches for quality management in federation. A selection process that can be used by cloud providers for collaboration is discussed in [20]. It discusses several pricing and QoS issues but no implementation has been done. Approach in [21] describes QoS aware composition architecture using evolutionary algorithm. It has an optimization component that allows QoS constraints. However energy efficiency of services is not considered in this approach. It is no clear from the implementation that how the mechanism would work in a federated environment. Dynamic resource pricing in the federation is introduced

in [15]. Only cost parameter is used as criteria and no other QoS is in consideration of this approach. Similarly in [22] discussed about price and reliability of services. Kertsz et al. [12] discusses approach for autonomous service provisioning in federation. They employed the idea of Global Service Registry that would contain service performance information. Depending on this information the broker chooses services. This approach does not consider energy efficiency as a parameter to judge the service abilities.

Another concern in SaaS provisioning and maintenance is service replaceability. The reason behind it is described in [7]. In [7] it is shown that structural, behavioral, and QoS induced changes to services may cause failures.[16] incorporates replanning of the service bindings when an failure occurs. A large deviation of QoS or violates a user constraint, then replanning approach works. This alternative replacement services are not available automatically. An adaptive replanning mechanism in [2] places more emphasis on the cause-effect relationship among services and accounts the service failures. An adaption framework and dynamic optimization strategies selection is the key factor in the approach. But replaceability in true sense has not been achieved in these two approaches. Finally a method in [2] was a significant one with respect to service replaceability issues in web. This approach replaces the critically failed components in a service composition plan. But this issue handles a few of QoS parameters that may not be sufficient to work in a vast domain like federated cloud. Considering all the above approaches for federation and service replacement, a new approach for service replaceability along with energy efficiency in a federated cloud environment is designed.

Scope of the work

It is clear from "State of the art" section that to assess service offers we need to see the price and the quality performances. In this regard, the competitiveness can be judged by a set similar services and Risky services (which service don't have any similar alternatives). While comparing business offers and providing the best possible solution to consumer, it is always a challenge to avoid harming the balance in federation. Replacement and negotiation are two main strategies for achieving this. These are helping in accommodating the user request into existing federation and prevent frequent re-federation. The cloud federation may be conceptualized as an Ecosystem and depicted in Fig. 1.

The Combined Service Pool, Request Queue and the Brokerage are the participating entities within the Ecosystem. The pool is a collection of the services provided by each CSP participating in federation. The request queue has been formed through just appending the request of the consumers. The brokerage plays the role of middleman and responsible for ideal service provisioning.

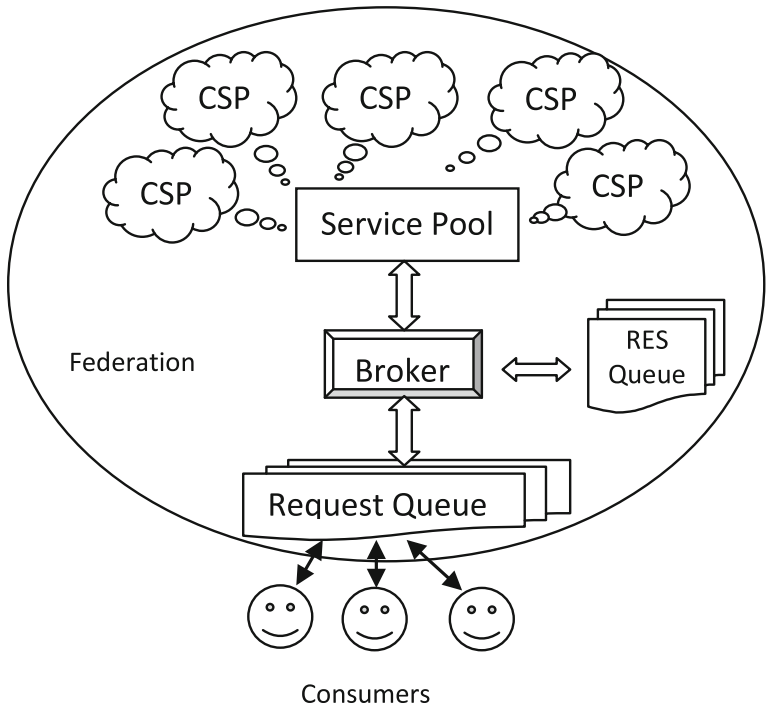

Fig. 1 Cloud federation

Ecosystem as defined in [18] signifies the producer consumer relationship. In general, there is a state, called Equilibrium that ensures the balance between the producer and consumers. The equilibrium state of the proposed ecosystem is achieved when the Request Queue is empty; i.e. all the requested services are served as per requirements. The loss of equilibrium may happen due to many reasons like when the already selected provider is failing to provide the service with desired QoS level. This deviation from equilibrium may trigger "Re-federation". The objective of the work is to make the federation autonomic in this context. The broker will take an attempt to bring back the equilibrium through well defined adjustment mechanism at earliest. As a result the re-federation may be avoided until the situation compels to do so.

There is a recent work [11] which takes care of inter operating cloud solutions. This approach in [11] tried to accommodate huge data in mobile devices exploring several IoT characteristics. This scenario invites some performance related concerns. For example, one user needs data storage service for his work backup. He has options as Google Drive, DropBox, etc. For this services concerned QoS parameters are Storage size, Response time, Availability. Even after subscribing for one service (say Google Drive here), the mentioned qualities may not seem good enough in some point of time. In this situation, the replacement mechanism would work to find the similar alternative matching the user's quality criteria. When any of the available alternate solution is not matching the quality requirement, the user may be offered to negotiate in some QoS(s). In both the cases, migration cost for shifting from one ongoing offering to a new one should be kept in mind.

Problem formulation

The problem clearly states that the service performances are degraded due to several reasons and those services are needed to be replaced by similar services to satisfy consumer. Moreover, there may be no exact matched service as per the consumer requirement. Then how to negotiate the consumer(s) with available services is another concern. The problem scenario is a collection of two sub-problems. The first one is the problem of replaceability. It demands a solution to exchange an identified service with degraded quality by another from available service pool. This leads to find out a perfect match for the required service from a possibly large set of similar services. The problem of service replaceability is essentially a search problem [2]. The second problem is mitigation and QoS negotiation (in case of unavailability of perfect match to replace). This can be seen as a special decision problem [23].

Service consumers express their requirements by query. It contains a set of functionality to be supported by offered services matching the quality requirement(s) specified in it. Let us consider the set of all services for a typical functionality f_i denoted by $S_i = \{s_{ij} : j \in I\}$ where j belongs to set of positive integers.Each service have a quality specification Q_{ij} for jth service in S_i. Quality specification within a query is $Q = \{q_k = req_k\}$where q_k is the k^{th} quality parameter and req_k is the value of that k^{th} quality parameter. The set of replaceable services is denoted by R.

Search problem formulation

The Replaceability problem can be formally defined as a search problem. Let us consider for a specific functional requirement all set of services as the search space. The initial state is the state where only the services with required functionalities were found and no service is added to R. That is at initial state $R = \{\}$. There are some restrictions in terms of QoS parameters declared by query. The Actions are defined to take arbitrary service from S_i and check if it follows the QoS restrictions in Q or not. The goal state comprises of all services from Si who allows th restrictions in Q are transferred to R. The search problem is solved here efficiently by reducing the search space by narrowing it with respect to the QoS restrictions given.

Function problem formulation

QoS negotiation between two services can be defined in terms of decision (negotiable or not). It signifies whether a service is acceptable or not with a deviated QoS as an alternate of required service. Let us also define a relationship as N i.e. negotiable. It is a binary relation among two services denoting that the corresponding services are negotiable with each other. Formally N is defined as $N = \{N_i : i \in I\}$. N_i is defined over S_i for a specific functionality f_i. i.e. $N_i \subset S_i \times S_i$. Let us consider $N_i(a, b)$ as a collection of Boolean formula m with variables as $x_1, x_2, ..., x_n$ ($x \in \{True, False\}$). Let us consider l_{q_i}'s are quality levels of q_i^{th} QoS. Each 'level' is defined as a range of permissible QoS values (Quantitative or Qualitative) mapped as a single qualitative value. Allowed levels for any q_i th QoS parameter are assumed as $l_{q_1}, l_{q_2}, ..., l_{q_n}$. The problem is to find an assignment of x_i's such that m is collectively True.

Service ecosystem and related term definitions

In this section some definitions are discussed that will be used in rest of the paper. A services Set (S) contains all the possible atomic services. S is defined as follows, $S = \cup_i S_i$ where S_i is the set of services as defined above. Each service s_{ij} has quality offerings declared by service providers as Q_{ij}. Q_{ij} is declared as set of quality parameters and corresponding levels as follows $Q_{ij} = \left\{q_k \mapsto l_{pq_k} \forall k \text{ and } p \in Z^+\right\}$. A query is denoted by Q which has a functional as well as a quality portion. Functional part of query describes the functionalities that are to be collectively satisfied by the service(s). Quality

requirements are specified by the following set as $q_k = req$ where q_k is the k^{th} QoS parameter. Query is always submitted to service broker that is mid entity between end users and cloud providers. Broker has two queues where it maintains the service request and service provisioning information. Service requests are submitted by providers are stored in the queue named REQ. Service request information already served by the broker are saved within a RES queue.

Ecosystem Ecosystem is defined with a set of services, service consumers along with broker(s). Here set of services has different service quality offerings and consumers have some service requirement. The ultimate objective of ecosystem is to provision required services by the consumers. The ecosystem changes its state(s) every time it receives newer request or older service adjustment to be done. The state of the ecosystem is represented by E_t for describing at a specific time point t. The possible state of the ecosystem are equilibrium and disturbed_equilibrium. There are a few components bu which the ecosystem is defined here. First of all the service pool is the main part of the ecosystem. It contains all services that are available for consumption. Two queues namely Request (REQ) and Response (RES) are stating the consumer requirement and status of ongoing services at any time. These two queues are used to decide the ecosystem states at any time.

Equilibrium It signifies a state in the whole ecosystem when the broker's REQ is empty and there is no intermediate request that is un-served. i.e. not added to the RES queue. Suppose, the status of the RES and REQ is stated by $RES_t = \{\text{served services at time t}\}, REQ_t = \{\}$. This typical state of the ecosystem is called equilibrium and the ecosystem at equilibrium is denoted by ξ_t. The equilibrium state of the ecosystem may be disturbed (named as disturbed_Equilibrium) due to the new service request and the degrading quality performances of the already provisioned ongoing services. That is at a specific time t, the ecosystem may be at ξ_t . But at time t' the ecosystem goes to the stat $E_{t'}$. Broker then take roles to restore equilibrium state $\xi_{t'}$.

Congruence Distance and **Congruence** Two services s_{ij} & s_{ik} are said to be at δ distance if s_{ij} & $s_{ik} \in S_i$ and QoS parameters are the same but differs in δ QoS parameter levels. Here δ is called congruence distance between s_i & s_j. Furthermore, two services s_{ij} & s_{ik} are said to be in δ - congruence relationship if s_{ij} & $s_{ik} \in S_i$ and QoS parameters are the same i.e. same QoS parameters are specified (i.e. Q_{ij} and Q_{ik}) though the subsequent levels are not the same. They differ in δ QoS parameter's levels. Say for example, s_{ij} & s_{ik} are two services with "payment" functionality from Bank domain. s_{ij} & s_{ik} are said to be congruent if QoS parameter Availability is mentioned in both the service description; but the may be s_{ij} is leveled as "high" & s_{ik} is leveled as "medium" available. These qualitative measurements of QoS parameter is done by clustering on the QoS values with same functionality domain. In this case s_{ij} & s_{ik} varies in 1- congruence distance (as there is one level distance between "high" and "medium").

Functional Similarity Two services s_{ij} & s_{ik} are said to be in functionally similar relationship if s_{ij} & $s_{ik} \in S_i$ but the subsequent QoSs may not match.

Absolute Similarity Two services s_{ij} & s_{ik} are said to be in absolute similar relationship if s_{ij} & $s_{ik} \in S_i$ and QoS parameters are same (i.e Q_{ij} and Q_{ik} are same) as well as subsequent QoS parameter's levels are also same.

Goodness Similar Selection of the services based on a few specific quality parameter may lead to dissatisfaction of consumers with respect to other important QoS parameters. It is important to provide services at moderate satisfaction for all QoSs. A method for such collective measures of QoSs are well described in [4]. It involves all QoS parameter to calculate the goodness of services. This measure is used here to calculate overall goodness of the services. Two services s_{ij} & s_{ik} are said to be similarly good s_{ij} & $s_{ik} \in S_i$ and QoSs are the same. i.e. (Q_{ij} and Q_{ik}) are same and also goodness values of s_{ij} & s_{ik} (G_{ij}, G_{ik}) are within a small threshold distance from each other. i.e. $|G_{ij} - G_{ik}| = Th$. Value of the threshold (Th) depends on the functional criteria of services. Sometimes $Th \leq 1$.

Goodness Distance Two services s_{ij} & s_{ik} are said to be at k-goodness distance if s_{ij} & $s_{ik} \in S_i$ and QoSs are the same i.e. Q_{ij} and Q_{ik} are same and goodness values of s_{ij} & s_{ik} (G_{ij}, G_{ik}) are within $\pm\theta$ from each other. i.e. $|G_{ij} - G_{ik}| = \theta$.

Emission Quotient Each of the atomic service (SaaS) is collection of some operations. Operations may be of computing, data access or other types. Based on the type of operations the carbon emission amount can be calculated. The total emission amount of a single service is defined as the Emission Quotient of the service. This is denoted by E_i and defined as follows.

$$E_i = \sum_j (e_j \times n_{ij}) \tag{1}$$

where e_j denotes the standard emission quantity for jth operation and n_{ij} is number of j^{th} operation in i^{th} service.

Solution proposal

In this section, we have presented the solutions considering the identified three sub problems as mentioned in "Introduction" section. The first problem has the solution in terms of service replaceability. It is a mechanism for replacing a service while it is a part of ongoing service allocation plan. This already provisioned service with degraded quality may be replaced by another service of similar quality on the fly (not or negligibly interrupting the provision).

Newer service requests can be fulfilled by any available service(s) whose quality criteria matches with that of the required service(s). In general, in real life broker provides a higher quality service to the consumers (more than requirement as specified in Q) in case there is no available service(s) that exactly matches the quality requirement(s). It may generate satisfactory performance but it is harmful for the proper utilization of service capabilities in terms of quality. Over-provisioning is often a common mistake or compulsion that happens. Many poor allotments of services may exhaust the full capacity of the federation. Broker has to build up a negotiation mechanism that would work in accordance with replaceability mechanism.

Replaceability and negotiation mechanism helps the cloud broker towards the optimized consumption of SaaS resources. Iterative application of these mechanisms gradually moves the federation towards saturation of its service provisions (mainly respect to the QoSs). Full exhaustion of offered services would result the federation to reconstruct further. This is called the process of re-federation. Often the federation advertises for newer service providers to join the federation based on the frequent requirements and availability of those services.

Service replacement on the fly

Service Replaceability is defined in [2] in terms of fitness and penalties. Violation from the QoS constraints increases penalty that in turn decreases Replaceability. But the process involves fewer QoSs parameters. In this proposed approach replaceability is measured in terms of all available QoSs and their corresponding constraints. The proper understanding of the replaceability and replacement algorithm needs some more terms to be described.

Congruent- Replaceable A set of all services that are in k congruence relation with s_{ij} are said to be congruent replaceable set for service s_{ij}. It is denoted by (CR_{ij}^{k}) where k can vary over Z^{+}. It is important to notice that lesser the value of k the more accurately replaceable is the corresponding set of services.

Goodness- Replaceable A set of all services that are goodness similar with s_{ij} are said to be goodness replaceable set for service s_{ij}. It is denote by GR_{ij}.

Absolute Replaceable A set of all services that are absolute similar with s_{ij} are said to be absolute replaceable set for service s_{ij}. It is denoted by AR_{ij}. It is also to note that 0-Congruent replaceable set is the same as Absolute Replaceable set. So, It is clear that $CR_{ij}^{0} = AR_{ij}$.

Functionally Similar Services A set of all services that are functionally similar with s_{ij} are said to be set of functionally similar service for service s_{ij}. It is denoted by FS_{ij}.

In the definition of service replaceability all of the above cases are aggregated with the relative importance it holds. It is clear that the set of absolute replaceable services are the most desired ones when we are thinking of replacing a service. So in the measurement of service replaceability it should carry the most weight. From the findings of [4] it is evident that the services with similar goodness often carry similar QoS performances. Congruent is the most compromised solution of service replaceabilities. The replaceability quotient for the replaceability of jth service in S_i is r_{ij} and is defined as

$$r_{ij} = \frac{||AR_{ij}|| + ||GR_{ij}|| + \sum_{k=1}\left(\frac{1}{k+1}\right)||CR_{ij}^{k}||}{||FS_{ij}||} \tag{2}$$

The more the Replaceability value of a service more the service is replaceable. Often, in some composition plans the candidate services having the least replaceability are called as critical component services. Replacement for those services is very difficult and often consumer has to compromise in some of the QoS parameters. The reduced problem of replacement after defining replaceability can be modified as a two objective optimization problem. Here objective function can be written as Maximize

$$Z = a \times r_{ij} + b \times \left(\frac{1}{E_{ij}}\right) \tag{3}$$

Subject to the QoS constraints on the existing ongoing service.

Here a and b are quotient which signifies the importance of the objectives. It can be tuned according to the application's need. If replaceability is the most important issue then a should be more than b. Otherwise a less than b. The service replacement algorithm is described in the following subsection .

Replacement

In the Replaceability algorithm that the steps for choosing the most replaceable and least emissive services. Individually these are interchangeable depending on the value of $a\&b$ in Eq. 3. If $a \geq b$ then replaceability step has to be executed first otherwise emissivity is first to decide. On going service demands as well as newer service requests both are submitted to the REQ. Broker finds services with previous service specifications and replaceability values. These are older service requests. Where as in newer request that replaceability quotient is absent. In case of newer request standard broker based discovery runs for initial solution to the query. Based on discovered services the replaceability quotient of that service is calculated. But often there is no existing available services that could be given readily. In that cases, the new requests are mapped into already provisioned services and a set of absolute similar services are found. Some of those absolute similar services may be replaced with lower quality services (found suitable either by replacement or negotiation) and freed.

The new request is served immediately with the freed service. Here Q is latest the query in REQ. The method of replaceability is discussed through the algorithm.

Algorithm 1 REPLACE

Input: $f_i, Q_{ij}, \theta, E_{ij}$ of the service $s_{ij}, S_{ij}^R = \{\}, T_i = \{\}, AR_{ij}, GR_{ij}, CR_i^k \forall \theta$

Output: Best Replaceable service set for$s_{ij}(S_{ij}^R)$ with maximum $r_{ij} and E_{ij}$

1: Initialize: Th=0.2
2: Search for the services in AR_{ij} such that $|r_{ij} - r_{ik}| = 0.2$ into the set T_{ij}
3: Search for the service having $E_{ik} \leq E_{ij}$
4: Save the service in S_{ij}^R
5: **if** $S_{ij}^R = \{\}$ **then**
6: Search for the services in GR_{ij} such that $|r_{ij} - r_{ik}| \leq Th$ into the set T_{ij}
7: Search for the service having $E_{ik} \leq E_{ij}$
8: Save the services in S_{ij}^R
9: **if** $S_{ij}^R = \{\}$ **then**
10: **for** $p = 1$ to θ **do**
11: Search for the services in CR_{ij} having $E_{ij} \leq E_{ik}$
12: Save the service in S_{ij}^R
13: **end for**
14: **end if**
15: **end if**
16: **return** The service (s) in S_{ij}^R as best alternative to s_{ij}.

Algorithm 2 NEGOTIATE

Input: Q_i, θ, Priority QoS list (Pr), $Ng = \{\}$(Set of services that are negotiable with s_{ij})

Output: Ng

Find the QoS constraint type for each QoS (q_x) parameter in Q_i.
for $y = 1$ to θ **do**
for $x = 1$ to n **do**
if q_x is upper **then**
if $q_x^i \leq q_x^y$ **then**
p_y++
else
r_y++
end if
else
if $q_x^i \geq q_x^y$ **then**
p_y++
else
r_y++
end if
end if
end for
if for every r_y parameters $q_r^y \in Pr$ **then**
s_{ij} and s_{ik} are not negotiable
else
s_{ij} and s_{ik} are negotiable
end if
Put s_{ik} is Ng
end for
return Ng

Service negotiation for newer request

Negotiation mechanism works among the provider's quality offerings and the new service requirement(s). When newer requests for service provisioning along with specific quality demand occur, broker at first tries to choose services that have higher replaceability and lower emission quotient. It may happen that there is no service available with the required level of QoSs according to the newer service requirement. In that case broker adjusts the existing over-provisioned services with the lower quality alternatives and provisions those services to the newer requester.

The negotiation algorithm is associated with replacement algorithm. The process of negotiation is initiated when the service discovery procedure cannot find the exact match of the requested service along with its quality requirements. This process has two parts. In the first part the required service (already (over) provisioned to another service provider) and proper replacement for that service are identified. If this replacement cannot be found then the newer service request is satisfied by a compromised solution with respect to some of the QoS parameters. In those cases, negotiation can be done among the goodness levels and less important quality parameters. The QoS constraint type referred in the algorithm is found in [2]. If a constraint is of upper type then it accept the lower values than the required on as a valid choice. Similarly if the constraint is of lower type then it accepts the higher values as valid choice. Response Time is a typical example of Upper constraint and Availability is a lower constraint parameter. The above two algorithms work often individually and in some cases collectively to regain the equilibrium within the brokerage system. The algorithm for regaining equilibrium uses either replacement, negotiation or both. The brokerage algorithm for regaining equilibrium is described as follows.

Experiments

In this proposed work the federation environment is simulated. A software as a service pool with 56 distinct functionalities are considered. Randomly generated queries ran on the simulated environment created by CloudSim

Algorithm 3 EQUILIBRIUM

Input: Quality requirement for newer request (Q_N) that cannot be served by existing available services, Quality requirement of provisioned services which have degraded quality (Q_E), Service allocation and necessary information about all services.

Output: Decision-State of equilibrium is regained or not

```
if Q_N is empty then
    call REPLACE (Q_E)
    if no replacement found then
        call NEGOTIATE(Q_E)
        if Ng is empty then
            Equilibrium cannot be achieved
        else
            Q_E=Null
        end if
    end if
else
    if Q_E is empty then
        call NEGOTIATE(Q_E)
        if Ng is empty then
            Equilibrium cannot be achieved
        else
            Equilibrium achieved
        end if
    end if
end if
```

(described in Table 1). It handles mainly IaaS services on top of which the SaaS federation was constructed. Queries were run consecutively. In this section the dataset preparation and experimental results are discussed along with some inferences drawn.

Implementation and dataset

Dataset according to functionality are prepared from QWS dataset available in [3]. Here, Nine standard QoS parameters for a total of 2508 service invocations are listed. Parameters are namely, Response Time, Throughput, Availability, Reliability, Successability, Compliance, Latency, Best Practices and Documentation. This dataset is considered as the combined service pool in the federation. Similar services are grouped according to Absolute replaceable, Goodness replaceable and Congruent-replaceable services as discussed before. Then the service similarity also measured for each of the group of services. Among those similar services, QoS values are clustered and included into levels (such as high, low, medium, etc.). Random carbon emission quotient per service is generated. Each of the saas is allocated to VMs upon request based on availability. Moreover, the IaaS parameters as Response Time, CPU per cycle and size of input/output data are monitored for each of the ongoing services when each new request is served. Predefined threshold are there. Overcoming those threshold indicates the degraded performance in corresponding SaaS (s) and replacement mechanism starts working. Each of the system level parameters are monitored at every new request arrivals to check the quality levels (signified by 9 QoSs) of the on going services. The system specification of the simulation environment is described in the Table 1.

Set of random queries are generated. These queries are considered as requirements from consumers. Each of these queries contains specific functionality with some QoS constraints (mostly qualitative constraints mentioned for majority of the mentioned nine QoS

Table 1 System specification

System parameters	Specification
Simulation engine	CloudSim-3.0.3
Operating system	Windows 8
Front end	Eclipse Juno
Virtual Machine Monitor	Xen
System architecture	x86
Number of data centers	5
Number of host	100
Number of VM	100
Number of cloudlets	300
Number of users	20
Types of workload	Random

Table 2 Experiment on federation

Specifications	Value
SaaS functionalities	56
Number of services	2508
Total Queries run	300
Involved QoS parameters	9

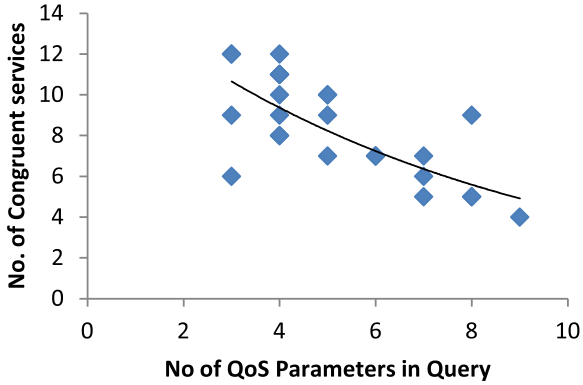

Fig. 2 Number of QoSs vs. Number of congruent services

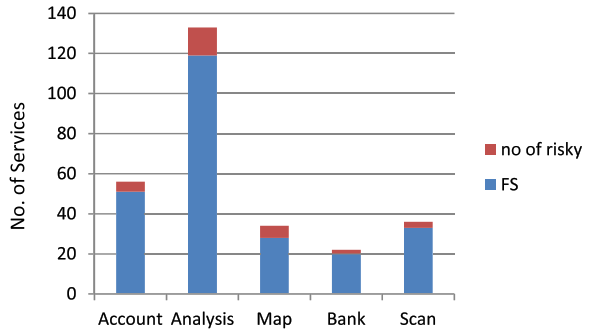

Fig. 3 Number of risky services per domain

parameters). Functionalities here are constrained to 5 (Account, Analysis, Bank, Map, Scan) domains. As a whole total 11 domains are involved in the pool. Each of the functionalities has multiple queries to run on. Each query runs replaceability as proposed as well as according to replaceability measure discussed in [2]. Moreover, each query is also run for proposed equilibrium gaining algorithm and other two algorithms discussed in [19] and [17]. The details of the experiment on federation is given in Table 2.

Results

There is an interesting relationship derived among the no of QoS parameters and number of congruent services in Fig. 2. It signifies that the number of congruent services is highly dependent on the number of QoS parameters specified. If the number of QoS parameters is less in the query then it is a possibility that there will be more number of congruent services.

Another analysis is being done for five functionalities. A service is pointed as a risky service if it has no congruent services. This signifies that the failure or over demand of those services may completely disturb the equilibrium and the federation will be broken. The Fig. 3 signifies that

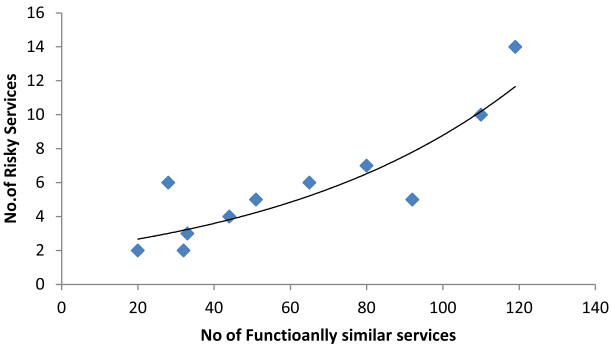

Fig. 4 Trend of risky services

Table 3 Outcome of 5 Queries in "Account" Domain

Total	Absolute	Goodness	1 Con	2 Con	3 Con	4 Con
7	1	2	4	2	0	0
7	1	1	2	3	1	0
9	1	2	3	3	1	1
12	2	3	4	5	1	0
6	1	1	2	3	0	0

"Analysis" services the highest set of functionally similar services as well as the number of risky services.

Another interesting inference can be drawn from the Fig. 4 that the number of risky services increases with the number of functionally similar services.

For five queries in "account" functionalities Table 3 is generated. Corresponding service replaceability quotient is also depicted in Table 4. It is evident from Table 4 that the services that have more congruence services are easily replaceable. Goodness similar services also affect the replaceability. More the available goodness similar services, more the replaceability obtained.

Comparison

The performance of the proposed approch is compared with respect to two aspects. Firstly the proposed approach is considered with respect to replaceability capability and compared with the approach in [2] . Then two self adaptation strategies as mentioned in [19] and [17] were compared with the proposed approach to signify how well the proposed approach can adapt with dynamic changes within federation.

Replaceability

Replaceability mechanism for service composition is implemented in [2]. We have compared proposed replaceability approach with that of [2] which considers fitness of GA as the key criteria while choosing services within a replaceable workflow. The comparative results are shown in Fig. 5. The replaceability Mechanism in [2] result in 19% matching executable services and 81% non-matching services. After employing proposed replaceability mechanism the Executable services were increased

Table 4 Replaceability values

Query	Replaceability
1	0.81
2	0.61
3	0.66
4	0.70
5	0.66

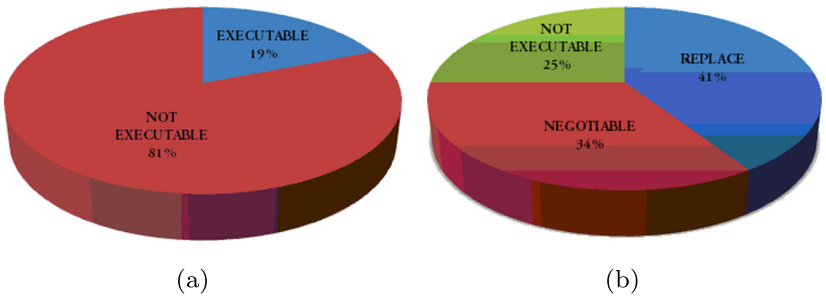

Fig. 5 Comparison among replaceability mechanisms. **a** Approach in [2]. **b** Proposed approach

41%. After introducing negotiation the total serviceable option increased to 74% (41% Replaceable + 34% Negotiable).

The replaceable set of services are calculated considering several different similar services. The percentage of Absolute Replaceable, Goodness replaceable and Congruent Replaceable services are 20%, 25% and 41% respectively. The result is depicted in Fig. 6.

In case of proposed Negotiation mechanism the services with 1 priority specification is higher achievable among the all other priority specification. This is depicted in Fig. 7 . Here the One specified priority QoS have a chance of 75% success while two, three and four priority QoS specification would have 59%, 46% and 40% success rate respectively.

Autonomous self-adaption

Thomas and Chandrasekaran [19] proposed a method of self adaptation in federation. How well this approach and proposed approach find out wide ranges of alternatives that have been listed. The number of satisfied services for each of the query run with proposed approach and the approach in [19] are noted. Same set of queries ran on the same service pool. Figure 8 signifies that the number of service alternatives is better in case of proposed approach. The services satisfied through the proposed approach have more alternatives compared to [19].

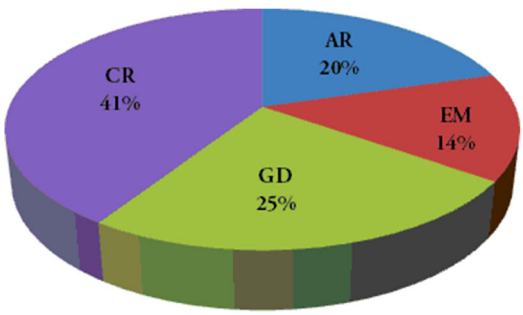

Fig. 6 Service types found in all executions

In Fig. 9, the service ecosystem has been observed for 300 queries (results shown for first 50 queries) on the same service pool with the proposed approach as well as the approach in [19]. The state of Disturbed equilibrium (possible re-federation) is attained more number of times for the approach in [19]. The proposed approach attained much less in disturbed_equilibrium state. It is evident that with the proposed approach the possible re-federation will be reduced.

Another recent approach on self adaptation as [17] is compared with the proposed approach. Number of service alternatives were listed for another random 50 random queries for [17] and proposed approach. Comparative performance is depicted in Fig. 10

Later [17] and proposed approaches were examined for an ongoing federation and checked how frequently both the approaches adapted with the changing environment. The results of equilibrium and disturbed equilibrium is depicted in Fig. 11. It is clearly shown that proposed approach (in Blue) adapts more appropriately when equilibrium gets disturbed.

The number of possible re-federation requests as generated by the proposed approach and approaches in [19] and [17] over 300 queries are depicted in Figs. 12 and 13 respectively. It is evident that the proposed approach invites less number of re-federation requests due to the imposed replacement and negotiation mechanism within the proposed equilibrium algorithm.

Conclusion

In this paper, the service ecosystem is defined for better service provisioning in cloud federation. Service offerings are here the producers and end users are the consumers of those service offerings. Often services meet the offered quality by the services provider at the beginning of provisioning. But often this is not maintained due to several reasons. Thus the poor performing services are needed to be replaced on the fly. On the other hand newer

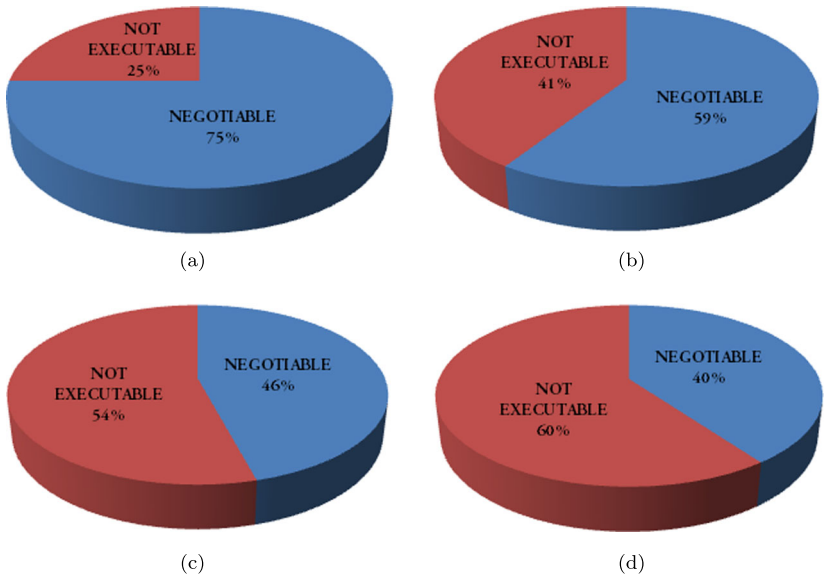

Fig. 7 Average number of services found for different restricted negotiation. **a** 1 QoS in priority. **b** 2 QoS in priority. **c** 3 QoS in priority. **d** 4 QoS in priority

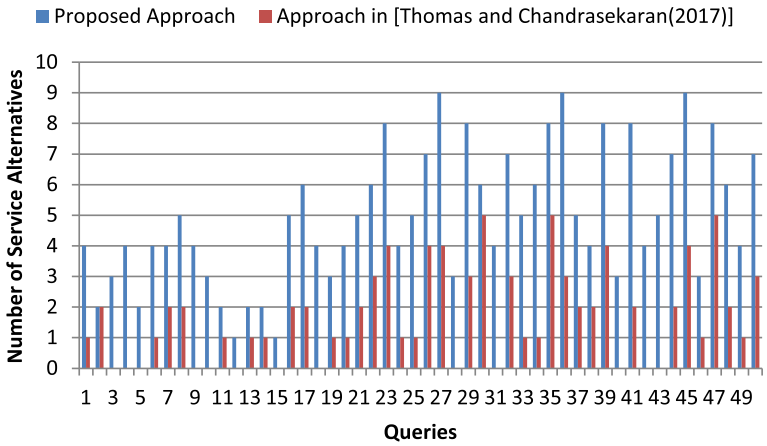

Fig. 8 Service alternatives in case of [19] and proposed approach

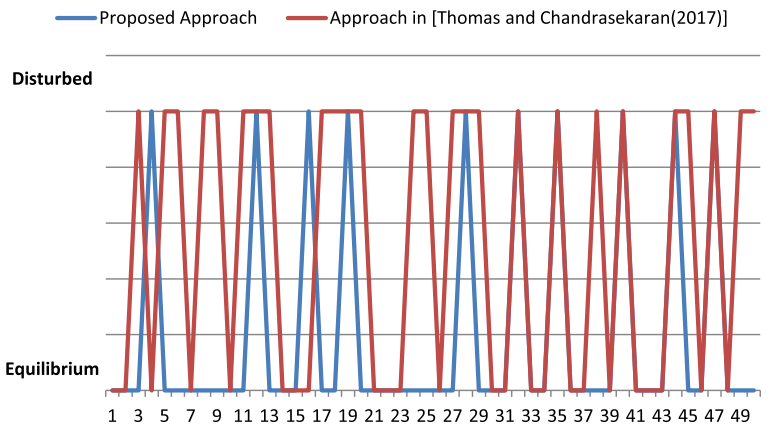

Fig. 9 State of equilibrium in case of [19] and proposed approach

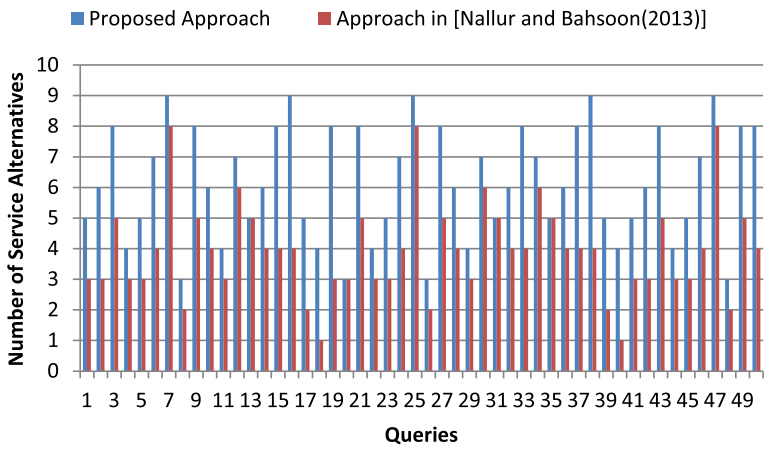

Fig. 10 Service alternatives in case of [17] and proposed approach

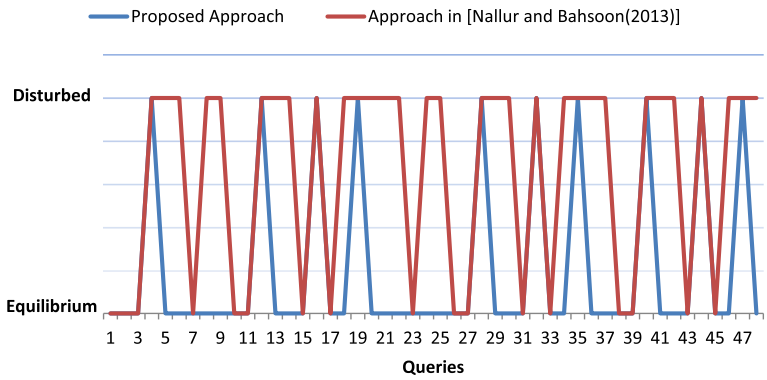

Fig. 11 State of equilibrium in case of [17] and proposed approach

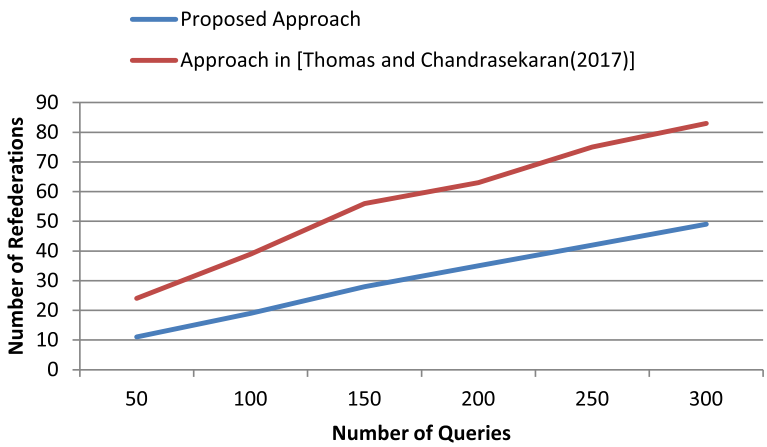

Fig. 12 Re-federation requests in case of [19] and proposed approach

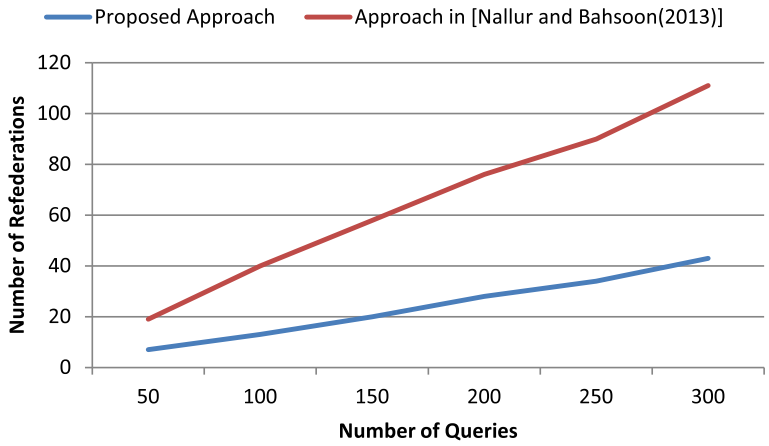

Fig. 13 Re-federation requests in case of [17] and proposed approach

service requests may have such a typical demand of service QoS levels that may not be possible to readily provide by the existing state of the ecosystem (federation). Here the replacement and negotiation are two mechanisms that help to regain the equilibrium in the ecosystem. Result shows that lesser the number of QoS parameters more the congruent services will occur. The risky points according to functionality are identified and it is shown that for analysis functionality the services have more risky components. The federation maintenance by mitigating the above issues will not only make the service provisioning better in quality and seamlessness but it will effectively increase the elasticity and scalability also. Services often in cloud are over provisioned due to the unavailability of services at a required level of QoSs. Replacement and negotiation of those over provisioning services help in increasing the elasticity and scalability indirectly. The proposed approach helped to solve three issues discussed in Introduction and as a consequence the frequency of re-federation is also reduced over a period of time. Future work includes the mechanism for insuring those services for guaranteed QoS levels as per requirement. Number of the service alternatives increase significantly by incorporating service replaceability. Equilibrium algorithm uses negotiation and replacement algorithm as per need to regain disturbed equilibrium within the ecosystem. Results show that the service ecosystem equilibrium is maintained more successfully for most of the incoming queries compared to recent approaches in [19] and [17].

Abbreviations

AR: Absolute replaceable; Con: Congruent; CR: Congruent replaceable; CSP: Cloud service provider; EM: Emission equivalent; GD: Goodness replaceable; QoS: Quality of service; SaaS: Software as a service

Acknowledgements

We are thankful to Ms. Soma Das M.Tech final year student of Department of CSE, University of Calcutta for her valuable contribution towards the proposed work.

Authors' contributions

All authors have participated in conception and design, or analysis and interpretation of the data, drafting the article or revising it critically for important intellectual content, approval of the final version.

Authors' information

Not applicable.

Funding

This manuscript is not supported by and research grant or fellowship.

Availability of data and materials

This manuscript uses data from the repository [3]. We have requested for the credential to download that data and was supplied with the same. This dataset contains service id with url and 9 QoS parameters for 2508 services. Later for the sake of implementation here we have attached some QoS values which were randomly generated.

Competing interests

This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue. The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript.

Author details

[1] University of Calcutta, Kolkata, India. [2] Università Ca Foscari Venezia, Venice, Italy.

Received: 30 March 2019 Accepted: 21 August 2019
Published online: 14 September 2019

References

1. Al Falasi A, Serhani MA, Elnaffar S (2013) The sky: a social approach to clouds federation. Procedia Comput Sci 19:131–138
2. Al-Helal H, Gamble R (2014) Introducing replaceability into web service composition. IEEE Trans Serv Comput 7(2):198–209
3. Al-Masri E, Mahmoud QH (2008) The qws dataset. http://www.uoguelph.ca/~qmahmoud/qws/indexhtml. Accessed: 12 Mar 2013
4. Bhattacharya A, Choudhury S (2016) An efficient service selection approach through a goodness measure of the participating qos. In: Proceedings of the International Conference on Informatics and Analytics. ACM. p 94. https://doi.org/https://doi.org/10.1145/2980258.2980451
5. Buyya R, Ranjan R, Calheiros R (2010) Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. Algorithms Architectures Parallel Process:13–31. https://doi.org/https://doi.org/10.1007/978-3-642-13119-6_2
6. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. Futur Gener Comput Syst 25(6):599–616

7. Canfora G, Di Penta M, Esposito R, Villani ML (2005) Qos-aware replanning of composite web services. In: Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on, IEEE. pp 121–129. https://doi.org/https://doi.org/10.1109/icws.2005.96
8. Celesti A, Tusa F, Villari M, Puliafito A (2010) How to enhance cloud architectures to enable cross-federation. In: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. IEEE. pp 337–345. https://doi.org/https://doi.org/10.1109/cloud.2010.46
9. El Zant B, Gagnaire M (2014) Dynamic break even pricing for cloud federation. In: Globecom Workshops (GC Wkshps), 2014. IEEE. pp 70–74. https://doi.org/https://doi.org/10.1109/glocomw.2014.7063388
10. Kertesz A (2014) Characterizing cloud federation approaches. In: Cloud Computing, Springer. pp 277–296. https://doi.org/https://doi.org/10.1007/978-3-319-10530-7_12
11. Kertesz A (2015) Interoperable data management using personal and infrastructure clouds. IEEE Cloud Comput 2(1):22–28
12. Kertsz A, Kecskemeti G, Marosi A, Oriol M, Franch X, Marco J (2012) Integrated monitoring approach for seamless service provisioning in federated clouds. In: Parallel, Distributed and Network-Based Processing (PDP) 2012 20th Euromicro International Conference on. IEEE. pp 567–574. https://doi.org/https://doi.org/10.1109/pdp.2012.25
13. Kohne A, Spohr M, Nagel L, Spinczyk O (2014) Federatedcloudsim: a sla-aware federated cloud simulation framework. In: Proceedings of the 2nd International Workshop on CrossCloud Systems. ACM. p 3. https://doi.org/https://doi.org/10.1145/2676662.2676674
14. Mell P, Grance T, et al. (2011) The nist definition of cloud computing. https://doi.org/https://doi.org/10.6028/nist.sp.800-145
15. Mihailescu M, Teo YM (2010) Dynamic resource pricing on federated clouds. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, IEEE Computer Society. pp 513–517. https://doi.org/https://doi.org/10.1109/ccgrid.2010.123
16. Na J, Li Gh, Zhang B, Zhang L, Zhu Zl (2010) An adaptive replanning mechanism for dependable service-based systems. In: e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on. IEEE. pp 262–269. https://doi.org/https://doi.org/10.1109/icebe.2010.68
17. Nallur V, Bahsoon R (2013) A decentralized self-adaptation mechanism for service-based applications in the cloud. IEEE Trans Softw Eng 39(5):591–612. https://doi.org/10.1109/TSE.2012.53
18. Olson JS (1963) Energy storage and the balance of producers and decomposers in ecological systems. Ecology 44(2):322–331
19. Thomas MV, Chandrasekaran K (2017) Dynamic partner selection in cloud federation for ensuring the quality of service for cloud consumers. Int J Model Simul Sci Comput 08:1750036
20. Wenge O, Siebenhaar M, Lampe U, Schuller D, Steinmetz R (2012) Much ado about security appeal: cloud provider collaborations and their risks. In: European Conference on Service-Oriented and Cloud Computing. Springer. pp 80–90. https://doi.org/https://doi.org/10.1007/978-3-642-33427-6_6
21. Wu CS, Khoury I (2012) Qos-aware dynamic research component composition for collaborative research projects in the clouds. In: Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on. IEEE. pp 883–888. https://doi.org/https://doi.org/10.1109/cloudcom.2012.6427536
22. Xin L, Datta A (2010) On trust guided collaboration among cloud service providers. In: Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on. IEEE. pp 1–8. https://doi.org/https://doi.org/10.4108/icst.trustcol.2010.6
23. Yu P (1973) A class of solutions for group decision problems. Manag Sci 19(8):936–946

Publisher's Note