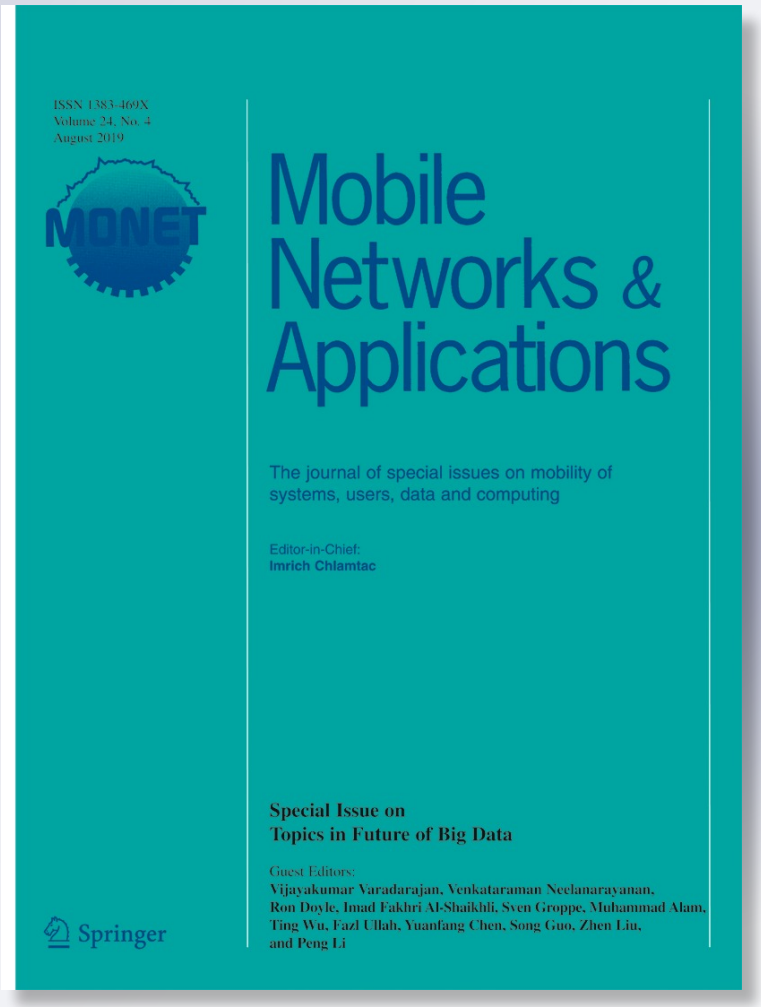
Efficient Algorithm for Identification and Cache Based Discovery of Cloud Services

Abdul Quadir Md, Vijayakumar Varadarajan & Karan Mandal

Mobile Networks and Applications
The Journal of SPECIAL ISSUES on Mobility of Systems, Users, Data and Computing

ISSN 1383-469X
Volume 24
Number 4

Mobile Netw Appl (2019) 24:1181-1197
DOI 10.1007/s11036-019-01256-0



Springer

Mobile Networks and Applications (2019) 24:1181–1197
https://doi.org/10.1007/s11036-019-01256-0

Efficient Algorithm for Identification and Cache Based Discovery of Cloud Services

Abdul Quadir Md[1] · Vijayakumar Varadarajan[1] · Karan Mandal[1]

Published online: 26 April 2019

Abstract
Efficient resource identification and discovery is the primary requirements for cloud computing services, as it assists in scheduling and managing of cloud applications. Cloud computing is a revolution of the economic model rather than technological. It takes advantage of several technologies that were tested and modified by replacing the local use of computers with centralized shared resources that are managed and stored by Cloud Service Providers (CSPs) in a transparent manner for Cloud Consumers (CCs). With this new use, various cloud services have appeared and it is mainly classified into three broad categories i.e., Infrastructure as a service (IaaS), Software as a service (SaaS) and Platform as a service (PaaS). Each of these cloud services provides several benefits to the CCs through their respective Quality of Service (QoS) metric. Among the cloud service models, most of the QoS attribute and metric are identical and some are different. The vendors of cloud have focused their objectives on the development of scalability, resource consumption and performance, other characteristics of cloud have been ignored. While CSPs face challenging difficulties in publishing cloud services that displays their cloud resources, at the same time CCs do not have standard mechanism for cloud resource discovery, automated cloud services selection, and easy use of cloud services. In this frame, this paper puts forward a set of QoS metric for SaaS, IaaS, PaaS services and propose (i) An efficient algorithm for identifying the cloud services based on the QoS metric given by the cloud consumer using decision tree classification algorithm (ii) An efficient algorithm for Cloud service resource registry which aims to enable CSPs to register their services with its QoS attributes and (iii) A Cloud service resource discovery that search for the suitable cloud service and their attributes in the cloud service registry that meets the CCs application requirements using Split and Cache (SAC) algorithm. Our new approach makes the provisioning of cloud service possible by ease of resource identification, publication, discovery based on dynamic QoS attributes via web GUI interface backed by series of test that has validated and the proposed approach is feasible and sound. The recommended solution is important: instead of putting effort in locating, learning about the services and evaluating them, CCs can easily identify, discover the services, select and use the required cloud resources. The efficiency of our algorithms was assessed through experiments using CloudSim, which primarily decreases the response time, CPU utilization and memory consumption for identifying and searching the cloud services and increases the accuracy of the CSPs list retrieved along with their QoS attributes.

Keywords Cloud consumers · Cloud services · Cloud service providers · Cloud service registry · Cloud service discovery · Quality of service · SaaS · PaaS · IaaS

1 Introduction

Cloud Computing signifies to hardware, software and data-centers accessible as a facility over a network and reachable via various computing devices such as computers, smart phones, etc. Although it is rather new computing model that has appeared in the last decade, it benefits from the ideas that have been proven, such as distributed computing [1], virtualization [2], web services [3], service-oriented architecture [4], grid Computing [5], etc. Cloud computing is surrounded within which CCs will access the storage, computing infrastructure and applications from anyplace and at any time over the internet. Cloud computing delivers its

✉ Abdul Quadir Md
abdulquadir.md@vit.ac.in

Vijayakumar Varadarajan
vijayakumar.v@vit.ac.in

Karan Mandal
karan.mandal2017@vitstudent.ac.in

[1] School of Computing Science and Engineering, Vellore Institute of Technology (VIT University), Chennai, Tamil Nadu 600127, India

resources as services in three broad categories. These services are classified into Software as a Service (SaaS), Infrastructure as a Service (IaaS) and Platform as a Service (PaaS). Each of these categories has specific characteristics and QoS attributes which makes it more suitable for certain groups of CCs. For example, companies are more likely to buy IaaS and PaaS services, where people will be more inclined to use SaaS services. SaaS permits CCs to remotely use the cloud applications that run on the cloud infrastructure by using compact clients. So, it doesn't require the CCs to invest in an infrastructure or purchase a software licenses.

For CSPs several costs are optimized through installation, deploying and maintenance of their services on cloud. Since various CCs access the same application. Some of the examples of SaaS are Google Drive, Salesforce, CRM, etc. [6]. PaaS offers a development environment as a service in which CCs can develop and implement their own applications. As a result, CCs need not administrate the infrastructure while maintaining management over the deployed applications and configuring the hosting environment. Examples of PaaS are Google App Engine, Salesforces Force.com and Microsoft Windows Azure [7]. IaaS offers basic storage and computing resources as a service, network equipment's, data-center, etc. These resources are used to run CCs applications. Usually, IaaS satisfies the best CCs requirements of interoperability and portability because they choose the various blocks that compose the infrastructure used. Some of good examples of IaaS are Amazon Elastic Compute Cloud and Microsoft SQL Azure.

The cloud computing preparation surroundings is classified into private, public and hybrid models. A private or non-public cloud is deployed on a private network and dedicated to a selected organization. Public cloud infrastructure is deployed and maintained by the CSPs and services area unit provided off-site above the internet.

Hybrid cloud is the mixture of private and public clouds. Some main features of cloud computing are availability, scalability, handiness, on-demand resource allocation and pay-as-you-use request mechanism [8]. A Public cloud [9] is provided by the immense organizations to general public. As similar to other model it can be also accessed via a network or the Internet. The fact about the public cloud is that it is transparent to all the CCs but still the data exchange and other services are confidential. A Private Cloud [10] is provided to the individual CCs of one organization that either accomplishes it or gives its controlling to a third-party. The main benefit of this deployment model is that there are no fixed limitations for security or bandwidth, as these resources are fully utilized by the organizations. A Community Cloud [11] is joint by organizations within the identical community. They can maintain their Cloud themselves or give the chore to a third-party.

A Hybrid Cloud [12] has two or more Cloud models cited above interrelated by standard or exclusive technologies. With the swift development of process and storage technologies and also the success of the web, computing resources have become cheaper, and a lot of powerful ubiquitously technologies are out there than ever before. This technological trend has enabled the replacement of computing model referred to as cloud computing. The rise of cloud computing has created an amazing influence on the data technology trade over the past few years, where massive corporations like Google, Amazon and Microsoft attempt to produce a lot of powerful cloud platforms, and business enterprises obtains them to reform their business models and take pleasure in this new paradigm. Indeed, Cloud computing delivers many compelling options that build it enticing to business homeowners [13].

The use of a cloud service presents several benefits for CCs. First, there is a great reduction in costs, since CCs buy only the resources they want, without surpluses or without spending money on maintenance or infrastructure. There is also the assurance of immediate and continuous access to storage and computing resources for any CC who has a machine connected to the network. In addition, CCs can adapt to the accessible resources to their particular requests and can increase resources as needed. All these benefits have increased the use of Cloud computing. Along with this increase, many new needs have arisen, and among them it is necessary to identify, discover and find cloud services that meet CCs requirements. Although there is elasticity in what CSPs are providing, the essential cloud resources are exclusive and dedicated for private uses. For e.g., whereas Amazon EC2 is most popular among IaaS providers, how its discovery and the selection mechanism of the required cloud resources from amid their virtual or physical servers is not known publicly or if such mechanism occurs at all. The objective of this paper focuses on how to discover the CSPs that satisfies the CCs requirements with minimal time complexity and reliable retrieved list of cloud services. For that, we put forward a complete system that carries out publication, identification and discovery of cloud services attaining the following aim:

- Introducing a Cloud Service Registry (CSR) framework that serves as a cloud repository for CSPs to publish and manage the QoS attributes of IaaS, SaaS and PaaS in a standard manner.
- Developing an identification system for cloud service model using decision tree algorithm based on different QoS parameters given by the CCs with minimal time complexity.
- Introducing a matchmaking algorithm for resource discovery by utilizing the QoS preferences of CCs using proposed Split and Cache (SAC) algorithm for quick and reliable retrieval of CSPs and their services from the cloud service registry.
- Evaluating the efficiency of the proposed approach using prototypical execution.

1.1 Paper organization

The rest of the paper is structured as follows: In Section 2, problem statement and related work in the area of cloud service publishing and discovery is presented. Section 3, introduces the QoS attributes for identification of IaaS, PaaS and SaaS services. Section 4, presents the proposed architecture of registry, identification and discovery of cloud services. In Section 5, the performance analysis and experimental results are discussed and Section 6, concludes the paper.

2 Problem statement and related work

2.1 Problem statement

In Cloud computing, cloud users and cloud services regularly collaborate with each other without having proper identification, registry and discovery mechanism. There is no such approach to identify the type of cloud service model based on QoS requirements of CCs and cloud applications. Often service registry and discovery challenges need to be solved for several reasons such as, cloud services not only provide data and business logic but also provide infrastructure functions at various levels. The Second issue is the standards for publishing and describing cloud services are missing. Like Web services that uses typical languages such as WSDL (Web Services Description Language) and USDL (Unified Service Description Language) to publish interfaces like UDDI (Universal Description, Discovery and Integration). The majority of cloud services being done are not established on description standards [14, 15] making the discovery of cloud services a difficult problem. Another major drawback is that of the cloud service discovery system, that it lacks automated cloud service identification model which should be done based on Cloud consumers requests and Cloud service discovery approach in the Cloud registry, based on QoS demands of the CCs with high accuracy and with minimal time execution.

2.2 Related work

The maximum usage of cloud service has led to the rise of new requirements, one of them is to have a standard system for identification model, publish, and discover the various cloud resources that meet CCs demands. Many approaches have been proposed by offering different approaches that helps the CCs to select an appropriate cloud services that meets their demand. It is somewhat diverse from the choosing the cloud resource components for composite intent, which is outside the scope of this paper. Our primary objective is to discover which CSPs along with their cloud services that matches the best requirements of CCs with minimal execution time, and not rank the CSPs from the selected CSPs to form final CSP that will be forwarded to CC.

Discovery of cloud services for consumers is considered to be an essential method in various domains such as peer-to-peer (P2P), mobile ad-hoc networks, service oriented computing and ubiquitous computing [15–18]. George et al. [19] surveyed on UDDI procedure for registering of services. Development established on UDDI has increased its reputation over the period of time. It is depended on message transfer using XML. While it is broadly accepted and executed, it is exceptionally static. The issue is that, resources are registered primarily but there state is not monitored dynamically. If a service which is registered is unavailable then no communication is sent to the CCs. Sun Service Registry [20] is Java utility based open source that uses ebXML and UDDI standards. It supports infrastructural maintenance and Web Service registry (WSR) for discovery of services. It is firmly incorporated into Java Enterprise System (JES), however there is no information about monitoring competence. IBM Web Sphere Registry (WSR) (http://www-01.ibm.com/software/integration/wsrr) adopts metadata repository for interaction of service using endpoints information. Kang et al. [21] introduces a portal for cloud attribute search engine. It uses the approach of similarity [22] and refers the agreed cloud ontology to choose the Cloud resources that match the CC requirements. Sim et al. [23] uses agent based computing to support the progress of software tools for cloud attribute administration, however QoS resources were not considered. Kourtesis et al. [24] introduces FUSION Semantic Registry which uses UDDI registry to find non-functional and functional features of the service. To find exact match between the request and response of the service discovery technique the authors adopts DL reasoning to publish their information.

Discovery of services is an agile area of research, specifically in field of Web services earlier, for services of cloud, challenges must be reassessed and results for efficient service discovery are limited. Few academicians have proposed the use of ontology methods for discovering cloud services. For e.g., Kang et al. [25] have suggested a (CSDS) Cloud Service Discovery System which uses ontology method to discover the cloud services that are nearer to CCs demands. Other academicians have proposed to use (DHTs) Distributed Hash Tables for improved load-balancing and searching of cloud resources. For e.g., Ranjan et al. [26] have put forward a cloud peer that encompasses Distributed Hash Tables to back the matching and indexing of range queries using multidimensional technique for cloud resource discovery. Goscinski et al. [27] proposes a cloud resource discovery and selection mechanism that uses a broker to match the cluster of cloud services. Zhou et al. [28] presents a P2P unstructured model for resource discovery in cloud. However, QoS attributes in their model is not considered. In [29] Zeng et al. introduces an

algorithm for selection of cloud services which determines the gains and cost of available resources that are accessed using proxy and returns those services that reduce the cost and increases the gains.

Zang et al. [30] proposes an algorithm for cloud service matching and composition and evaluates the semantic similarity between them to determine which of the two specified cloud resources are interoperable. L. Sun et al. [31] conducted the detailed survey on selection of cloud service methods which includes Multi Criteria Decision Making approaches such as MAUT [32], Analytic Network Process (ANP) [33], Analytic Hierarchy Process (AHP) and ranking techniques [34]. In [35] Godse et al. presents an Analytic Hierarchy Process technique for selecting the SaaS resources. The attributes used for selection are cost, vendor reputation, usability, architecture and functionality. The issue is that it takes only three SaaS cloud services for testing and not considering other categories of cloud services. Karim et al. [36] proposes a mapping method for QoS attributes by combining IaaS and SaaS services and ranking them using AHP approach for CCs. Eight attributes and four services where used for testing two categories of cloud services only. Silas et al. [37] proposes a middleware for cloud service selection using ELECTRE. It uses QoS attributes like the approaches cited above. ELECTRE ranks the resources to match the CCs preferences and needs. Menzel et al. [38] proposes an approach for selection of cloud services using Multi Criteria Comparison Method. This method uses ANP technique which allows the CCs to choose the IaaS services such as reliability, flexibility, etc. This model works only for IaaS providers.

Limam et al. [39] presents MAUT approach for selection of SaaS resources using cost, quality, and reputation which is reduced to single attribute feedback. A. Li et al. [40] proposes a CloudCmp architecture for comparison of cloud services. However their framework does not take account of complete cloud resource selection. In [41], selection of IaaS cloud resources is formalized using Multi Criteria Decision Making (MCDM). James et al. [42] proposes a QoS framework which intended for SaaS Enterprise Resource Planning (ERP). The model uses MCDM approach to recommend an appropriate SaaS ERP to the CCs. However, characteristics of cloud were not taken in account for selection procedure of both [41, 42]. Kanagasabai et al. [43] proposes semantic cloud resource discovery and selection using OWL-S. However, quality of service attributes was neglected.

From the above discussed related work it is clear that various efforts in literature exist that allows CSPs to publish and discover their cloud services. However the following issues needs to be addresses, (i) there is no complete system that governs the identification of IaaS, SaaS and PaaS cloud services based on their respective QoS attributes (ii) lack of standardization for publishing the cloud resources in the cloud service registry by the CSP, which demands a complete attention and (iii) while discovering the cloud services, cloud users QoS requirements and their respective values have been ignored which results in inaccurate retrieved list while discovering the cloud resources. Therefore, this paper put forwards QoS attributes for each of the cloud services to identify the cloud service model. Moreover, it introduces a standard cloud registry system for publishing, identification of the cloud services and an efficient discovery of the CSPs with minimal time complexity using SAC Algorithm along with the cloud users requirements and their QoS values. The applicability of the proposed system has been validated by using synthetic cloud dataset with experimental results. The use of synthetic data allows us to observe the performance of the method accurately.

3 QoS Attributes for IaaS, PaaS and SaaS

The primary entity with which CSPs communicate in a cloud environment are cloud consumers. These consumers obtain an increasing range of services namely IaaS, PaaS and SaaS from CSPs. In this section, we suggest the key QoS attributes and their various sub-attributes to identify each of the service model based on CCs preferences. However, few of the QoS attributes are similar for all the cloud service models but the approach we custom them are diverse for different cloud service model. In the proposed approach, CSPs register their QoS attributes based on service model in the cloud registry, therefore identifying these attributes and there model is of main concern which in turn reduces the response time for searching the resources in CSR when identified. These QoS attributes functions as a repository through the publication, identification and discovery process.

3.1 Attributes for infrastructure as a service (IaaS)

In order to select the desirable QoS metric over which IaaS service model can be identified and compared among different CSPs [44], a review was examined on latest research area. Selectively 27 Service Measurement Index (SMI) [45] QoS attributes were extracted as our identification and evaluation of IaaS service model. These SMI metric are classified into six top level attributes of quality of service group i.e. Accountability, Assurance, Agility, Cost, Performance and Security. SMI is an agreed set of key performance indicators that offer a regularized approach for identification, comparing and measuring of cloud services. It is a standard approach to support organizations evaluates cloud service established on specific QoS requirements and it is established by (CSMIC) Cloud Service Measurement Initiative Consortium. Figure 1 shows the QoS attributes for identification of IaaS provider established on SMI.

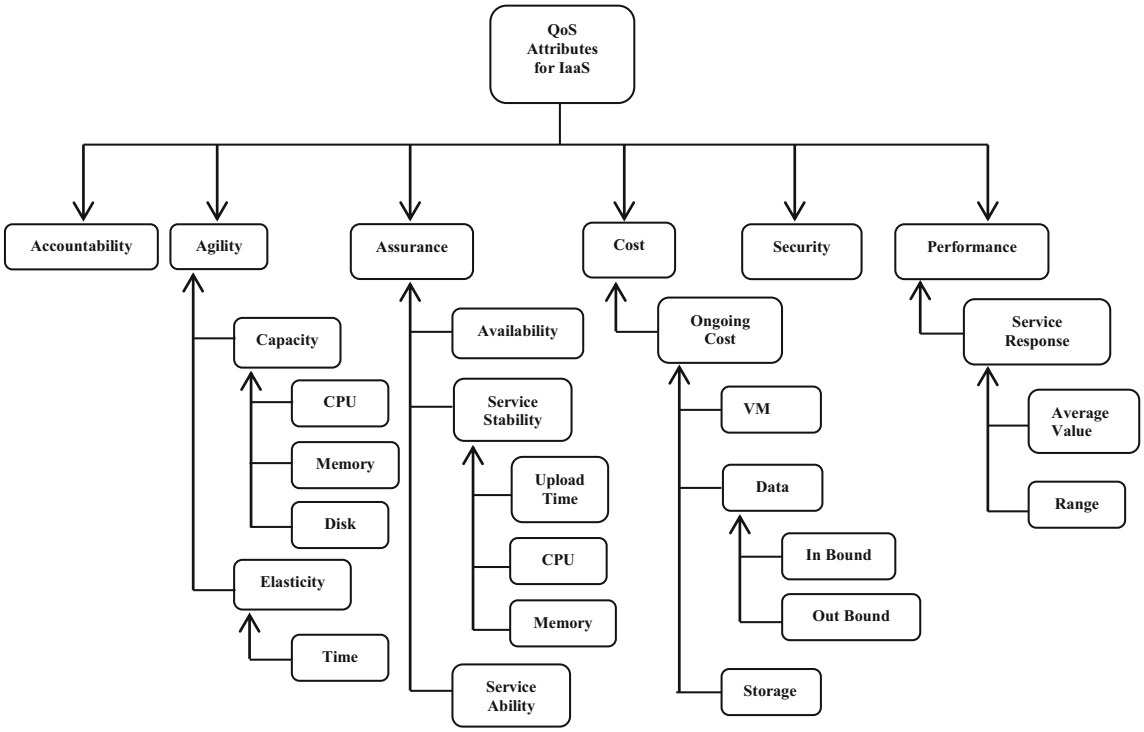


Fig. 1 QoS attributes for identification of IaaS service model

a) *Accountability*- this metric measures the CSPs diverse characteristics. This metric is decisive in establishing trust between CSPs and CCs. Organizations do not deploy their essential applications and store the data in a spot where accountability is missing.
b) *Agility*- the utmost vital benefit of cloud is that it changes and adjusts quickly at a lesser cost. It evaluates agility by how quickly a system adjusts to an environment if few variations are done by cloud organizations. First level attributes of agility are capacity and elasticity and second level attributes are cpu, memory, disk and time.
c) *Assurance*- this metric shows how the service providers execute and delivers the services to the CCs as established in SLA. Individual organizations want to increase their initiative and give exceptional cloud service to their consumers. Therefore, first level attributes like availability, service stability, service ability and second level attributes upload time, memory, CPU, free support and type of support are crucial factors in choosing the services.
d) *Cost*- whenever organizations want to deploy their application on the cloud, it checks whether it is cost effective or not. Hence for business and IT this attribute is the most crucial and it is biggest measurable attribute today. First level attribute is on-going cost and its sub attributes are VM cost, data and storage.
e) *Performance*- numerous different resolutions are presented by CSPs stating the IT requirements of several organizations. Each explanation has distinct performance centered on first level attribute service response time and second level attributes average value and range. All organizations must know in what way their demands are functioning on several clouds and if these requirements suit their views.
f) *Security*- data safety and security are decisive factors for all organization. Deploying ones data under third party providers is a critical concern which requires strict rules managed by CSPs. For instance, financial business needs compliance and regulations requesting data integrity and privacy. This attribute is multi-dimensional and contains metric like privacy, availability and data integrity.

3.2 Attributes for software as a service (SaaS)

Figure 2 shows the QoS attributes which are extracted from the recent research survey [46–49] for the identification of SaaS service model. Thoroughly 25 SaaS attributes were selected for our evaluation model. These SaaS metric plays a crucial part in establishing the QoS.

a) *Performance*- since SaaS is delivered by means of service over internet, performance is considered a primary QoS indicator to the CCs. Performance guiding principle are well-defined as service results in the

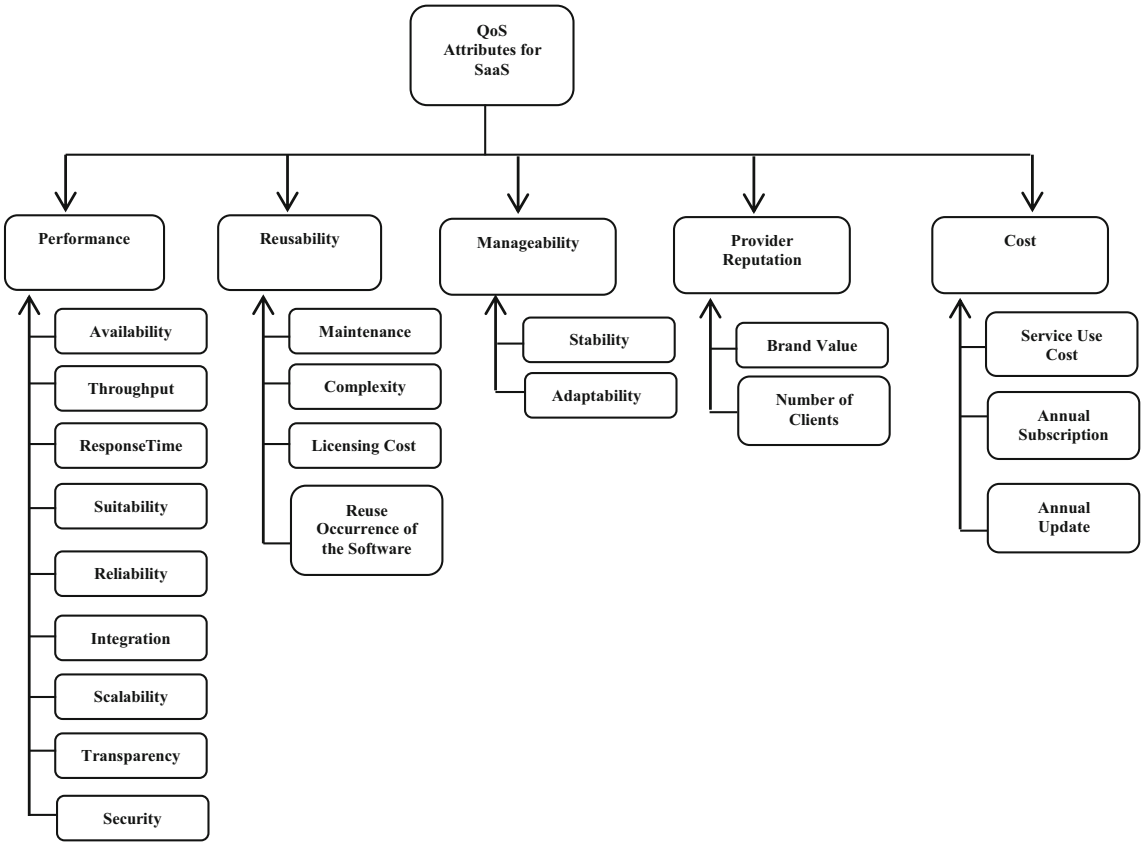


Fig. 2 QoS attributes for identification of SaaS service model

SLAs. These principles are essential for retention and CCs satisfaction. QoS attributes for performance are availability, throughput, response time, suitability, reliability, integration, scalability, transparency and security.

b) *Reusability*- this attributes defines how IT companies uses the software and how frequently the software meets the demands of the CCs. Since service providers buy and manage the software on their side, evaluating this attribute becomes at most essential. This feature intensely impacts return on investment made by the CCs. QoS attributes are maintenance, complexity, licensing cost, reuse of software.

c) *Manageability*- it is stated as ease through which cloud services can be managed when some changes are done with respect to changing demands and enhancements. Quality attributes are stability and adaptability.

d) *Provider Reputation*- this attribute is based on the feedback given by the CCs based on quality of service it provides to its customers. The sub attributes are brand value and number of clients.

e) *Cost*- this attribute is an on demand service, expenditures are computed on the source of how CCs utilizes the cloud service. Primary attributes which defines the SaaS cost are service use cost, annual subscription and annual update cost.

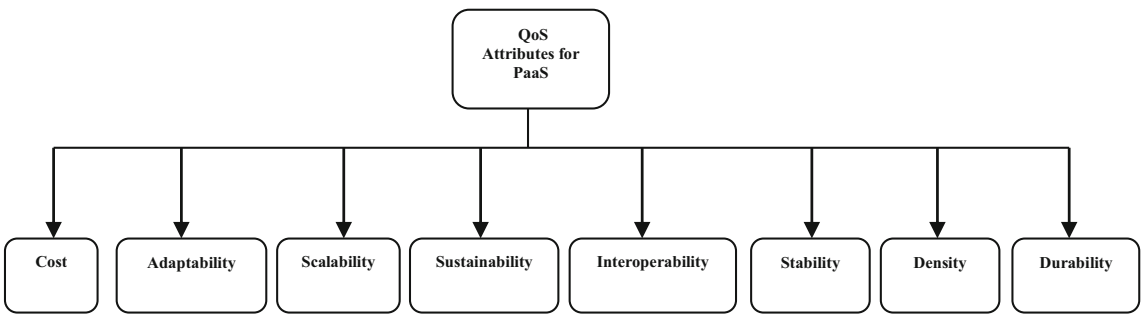


Fig. 3 QoS attributes for identification of PaaS service model

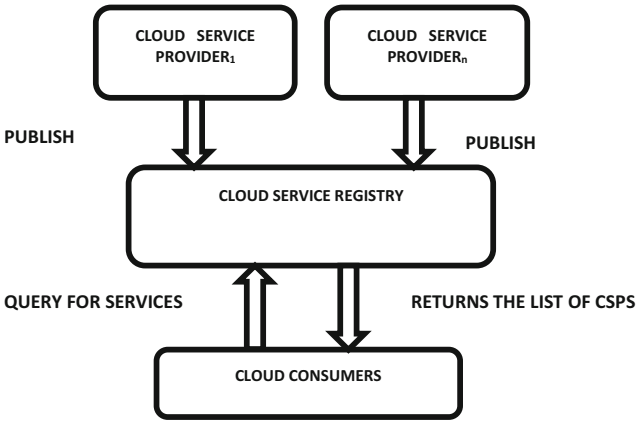


Fig. 4 Overview of Service Cloud Discovery

vider or different service provider. This attribute is of qualitative type and demonstrates in what way software services can be utilized with others.

e) *Stability*- is described as degree of uncertainty in performance of cloud service. In PaaS, it signifies degree of uncertainty in carrying out the platform instance.

f) *Density*- this attribute measures exactly how several occurrences of workload can execute and run on system under test before the system performance downgrades lower than a defined QoS.

g) *Durability*- denotes to protecting the data for long term, the data stored through PaaS must not undergo degradation, bit rot, or corruption. Durability is crucial for long and short term business attainment.

3.3 Attributes for platform as a service (PaaS)

PaaS is a computing framework that provides the CCs with an interfacing language, so that the SLAs and program logic can be expressed. To identify the QoS attributes of PaaS, we have selectively extracted the followings attributes for our evaluation model [50]. Figure 3 shows the QoS attributes for PaaS.

a) *Adaptability*- this attribute adapts to increase in the workload as well as decrease in the workload by allocating and deallocating the services in an automatic manner.

b) Scalability- this attribute adjusts only to increase in the workload by allocating the services in an incremental fashion.

c) *Sustainability*- if cloud service platform is used for a long term then this attribute plays a crucial role. It directly emulates the endurance of software. Data center performance, carbon foot print and power usage effectiveness are key attributes which plays an essential role.

d) *Interoperability*- it is capability to use further cloud services which may be provided by the same service pro-

4 Proposed cloud service registry and discovery system architecture

Locating the cloud services in an immense scale, composite network is an essential task. There can be numerous CSPs with distinct characteristics offering a diversity of various levels of CCs requirements. To create alertness regarding the availability and the status of these cloud services, a Service Cloud Discovery (SCD) method must be deployed. All CSPs will necessitate the registration of cloud services along with other required resources such as service model, its associated QoS attributes and values, so that CSPs services are broadcasted to the CCs. The information resource then binds all these services available to possible CCs, by applying Split and Cache matchmaking Algorithm (SAC) based on QoS request by the CCs with the cloud resources which are available and returns all the CSPs which have satisfied the CCs requirements. Figure 4 shows an overview of Service Cloud Discovery mechanism.

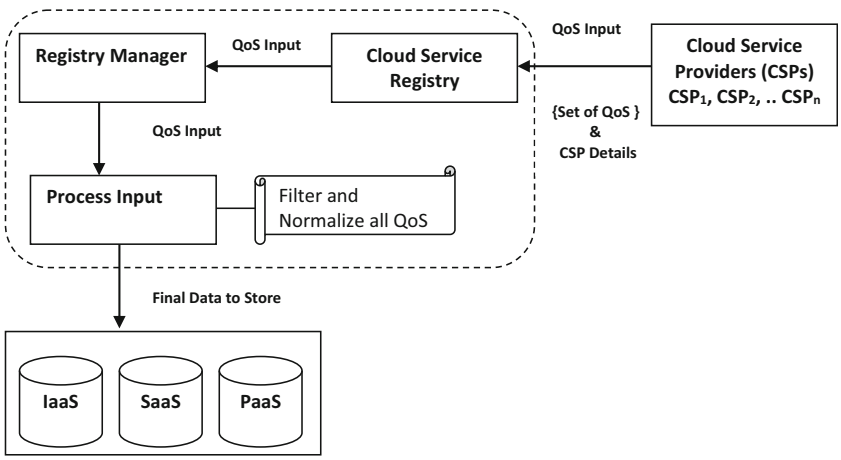


Fig. 5 Cloud Service Registry System Architecture

4.1 Cloud service registry system

The proposed Cloud Service Registry System (CSRS) is designed for CSP's to provide them a convenient and standardized way to register and publish their different kind of services, so that CC's can find and use the cloud services. Different CSP's may have different kind of services models and their associated QoS attributes. Whenever a CSP comes to register its services with the CSRS, they first have to provide all the basic information related to the their company and the service model along with their QoS attributes they are publishing, then all the information's related to the service with its attributes are stored in CSR. The registry system is capable to accept different attributes for all the three type of CSP's namely IaaS, PaaS, SaaS. The registry system shown in Fig. 5 consists of two major components 1. Cloud Service Registry 2. Registry Manager. CSP's only interacts with Cloud Service Registry module where they provide all the basic to advance information related to the company and service they want to publish. Once they provided all the information the input given is further process by registry manager and added to the dataset according to their cloud model type. Before storing them all the data is filtered, classified and normalized into correct format so that they can be stored properly. Algorithm 1 (named as CSR) presents the proposed cloud service registry for different service models. The Cloud service registry system consists of three major steps:

1. *Taking input*- Input is taken from the CSPs in two steps through the web Graphical User Interface (GUI).

a) Collecting all the information about the company or the providers.
b) Gathering all required information along with QoS Attribute and values for registering service with its service model type shown in the (Figs. 1, 2 and 3).

2. *Filter and Normalization*- Once the inputs are given by the CSPs, normalization is applied to all QoS attributes and its values as per their numeric, range, boolean or unordered data types.

a) Numeric or Range value – A numeric input can be a number or a range value with hyphen (−) separator that indicates the lower and upper limit.
b) Boolean value – A Boolean value can be of two types either True or False. Here the replacement of True and False is done with Yes and No.
c) Uncategorized value – An uncategorized value can be a single or multiline string provided by the CSP.

3. *Storing it to database* - At last step one row is created in dataset that includes all the information given by the CSP, and further it is added to the dataset according to its service model type which was taken from the CSP in step1 of CSR. The CSR Algorithm 1 works in six steps:

Step 1. (lines 1–3): The global variables are declared required by the algorithm to work. The variable DatasetDir holds the location of data sets – {IaaS, PaaS, SaaS}. Whenever algorithm needs to read or write into datasets, this location is used to reach and load the dataset.

Step 2. (lines 4–21): The CSR algorithm is defined between line 4 to 21 as a function name AddCSPRecord. This method is called with one parameter DataRow that holds all the input taken from the CSP in mapped format Key Value Pair (KVP), where Key is the name of the attribute or label and Value holds corresponding value to key.

Step 3. (lines 5–6): In line 5, Cloud model {IaaS, PaaS, SaaS} is taken from the input DataRow and assigned into the variable cm. In line 6, dataset is loaded with the help of OpenDataset() method that takes one string argument where value DatasetDir + cm is passed and then the loaded dataset is assigned to variable ds.

Algorithm 1 - Cloud Service Registry

```
        dr - Input (Mapped attribute and value)
        taken from the CSP
1.      global
2.      | DatasetDir = "Dataset/"
3.      end global
4.      function addCSPRecord(dr)
5.      cm = dr.Field("Model")
6.      ds = OpenDataset(DatasetDir + cm)
7.      rr = fg
8.      for attr in dr do
9.      av = "0"
10.     if isNumeric(attr) then
11.     av = NumberFormat(attr)
12.     else if isBoolean(attr) then
13.     av = BooleanFormat(attr)
14.     else
15.     av = UncategorizedFormat(attr)
16.     end if
17.     rr.add(av)
18.     end for
19.     ds.Append(rr)
20.     ds.Close( )
21.     end function
```

Step 4. (line 7): It declares and initiates variable rr (ResultRow) as empty set which is later used to add filtered values.

Step 5. (lines 8–18): In this, foreach loop, all KVP entry is iterated, for every iteration, single entry is taken and its data type is identified and filtered accordingly and then added to empty set rr.

Step 5.1: (line 10): Numeric type is checked with the help of method IsNumeric(), if it is a numeric data then it is formed to number format with the help of NumberFormat() method.

Step 5.2: (line 12): Boolean type is checked, if it is identified as the type of the attribute is Boolean then it is formed to our Boolean representation in yes or no format.

Step 5.3: (line 15): If the data is neither numeric nor Boolean then it is treated as uncategorized type, method UncategorizedFormat() is used to format it. After the identification and normalization of the data entry,it is added to the set rr.

Step 6. (line 19–20): The set rr is added to the dataset ds with help of Append() method. Once it is added in dataset is saved and closed with the new record by calling the Close() method on ds.

4.2 Cloud model type identification

The Cloud Model Identification (CMI) algorithm works with Decision Tree to classify the Cloud model type namely IaaS, SaaS and PaaS. Decision tree is a supervised learning that is used both for the regression problem and for the classification problem [51]. The proposed algorithm uses decision tree to classify Cloud model type from the given QoS attributes by the CCs. In this classsification, the main tasks is to identify the element intended for root node in each level, this method is called attribute picking. Two standard attributes picking measures for this are Information gain and Gini Index [52]. A decision tree classsification is a tree arrangement like a flow diagram, where individually internal node signifies a test on an attribute, individually branch indicates a result of the trial, the class label is characterized by separate terminal node or leaf node as shown in Table 1.

The primary algorithm for creating decision tree is called ID3 which adopts a top-down greedy approach to examine through the probable branches with no backtracking. The ID3 algorithm uses Information Gain and Entropy to generate a decision tree [53]. To identify Cloud model type, we have example frequency table shown in Table 2, which has only 7 attributes to illustrate the working of CMI algorithm with decision tree. Originally the frequency table holds many different attributes and their respective combinations that help to predict the cloud model type as shown in Table 3.

4.2.1 Entropy

It uses entropy to compute the likeness of a sample data. If the entire sample data is similar, then the entropy is 0, otherwise sample data is equally divided. To create a decision tree, two different types of entropy from the frequency tables is calculated.

Entropy of one attribute

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Entropy of two attributes

$$E(T, X) = \sum_{c \epsilon X} P(c) E(c)$$

4.2.2 Information gain

The gain of information is based on the decrease of entropy after a set of data is divided into an attribute. The construction of a decision tree consists of finding an attribute that returns the greatest information gain. For every iteration, information gain is calculated on dataset and dataset is further divided into sub datasets. Steps to calculate information gain and building decision tree is shown below:

1. Calculate entropy of the target dataset.
2. Split the dataset based on different attributes.
3. Pick the attribute with highest information gain as the decision node.
4. A branch has entropy zero is a leaf node.
5. A branch has entropy greater than zero needs further splitting.
6. Run the ID3 algorithm recursively until all non-leaf branches are classified.

Once all the training data is classified into leaf and pure subsets, decision tree is generated into the internal memory. The generated model is used multiple times to predict desired result. Further testing data is applied, the testing data is taken from the CC as a mapped QoS attributes and further passed to the CMI algorithm to predict cloud model type.

Table 1 Frequency of One Entropy QoS Availability

Cloud Model		
IAAS	PAAS	SAAS
5	4	3

Table 2 Frequency of QoS Availability

Accountability	Security	Customizability	Interoperability	Adaptability	Portability	Manageability	Type
YES	YES	YES	NO	NO	NO	NO	IAAS
YES	YES	YES	YES	NO	NO	NO	IAAS
YES	YES	NO	YES	YES	YES	NO	PAAS
YES	YES	NO	YES	YES	YES	YES	SAAS
YES	YES	YES	YES	NO	NO	YES	IAAS
YES	YES	NO	YES	YES	NO	NO	IAAS
YES	YES	YES	YES	YES	YES	NO	PAAS
YES	YES	NO	NO	YES	YES	NO	PAAS
YES	NO	NO	YES	YES	YES	YES	SAAS
YES	YES	NO	NO	YES	NO	NO	PAAS
YES	NO	YES	YES	YES	YES	YES	SAAS
YES	YES	NO	YES	YES	YES	NO	IAAS

4.2.3 Decision tree to decision rules

Decision tree shown in Fig. 6, Illustrates how decision tree is generated, and decision is made with QoS attributes and their availability for different Cloud model type.

Condition 1:
IF (Adaptability == NO).
THEN Cloud Model = IAAS.

Condition 2:
IF (Adaptability == YES && Manageability == YES) THEN Cloud Model = SAAS.

Condition 3:
IF (Adaptability == YES && Manageability == NO) THEN Cloud Model = PAAS.

The CMI Algorithm 2 has the following models:

Load or create model Whenever the CMI algorithm is invoked, it looks for the existing trained model, if it is found then the model is loaded in to the memory and further it is available to use for next N times until the system goes down. In case it fails to find or load the existing trained model, the new decision tree model is created and stored physically.

Table 3 Frequency of Two Entropy QoS Availability

		Cloud Model			
		IAAS	PAAS	SAAS	
Customizability	YES	3	1	1	5
	NO	2	3	2	7
					12

Predict cloud model type In this step, predict method of decision tree is called with decision tree model and QoS attributes. QoS attributes are passed to the CMI algorithm as a mapped value (Key value pair), where key is the label for the attribute and value field contains either true or false, true value indicates that user wants that service and false is vice versa.

Algorithm 2 - Cloud Model Identification

```
Input:
    p - QoS Attributes taken from CC (Test dataset)
Output:
    i - Predicted cloud model type
1.  function predictModel(p)
2.      m = NULL
3.      if hasModel("trained model") then
4.          m = LoadModel("trained model")
5.      else
6.          m = BuildDT("training dataset")
7.      end if
8.      i = DT .Predict(m, p)
9.      return i
10. end function
```

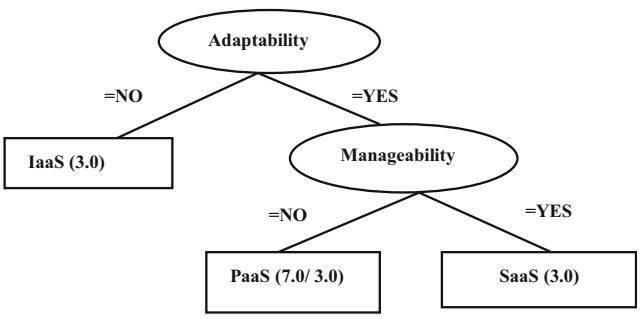


Fig. 6 Decision Tree Generation

The CMI Algorithm 2 works in 4 steps:

Step 1. (lines 1–10): The function PredictModel() is defined between line 1–10, this method is called with one single parameter p which is mapped set of QoS attribute and Boolean value True or False.

Step 2. (line 2): This declares and initializes a temporary variable m, which is later used to hold trained decision tree model.

Step 3. (lines 3–7): In the line 3 existence of trained model is checked, if it is found then in line 4 it is loaded and assigned to variable m, otherwise in line 6, new model is generated with the help of training data set and assigned to variable m.

Step 4. (lines 8–9): In the line number 8 Predict() method is called with decision tree model and users input (QoS attributes), once the execution of this method is completed predicted model is assigned to variable i, which is return to the caller method.

4.3 Cloud discovery system

The discovery system is for CCs (Cloud Consumers), where they can find CSP's that provides and full-fills all the service requirement of their application. The discovery system requires QoS as an input from the CCs to find all the matching CSP's registered with the CSRS. The Discovery system can search through the dataset containing thousands of records of CSP's offering different kind of QoS services and fetch all the CSPs according to the service model along their respective values with minimal time complexity. As an output, Discovery system serves all the CSP's matching the CCs requirements.

The discovery system architecture shown in Fig. 7 consist of the following components, 1. Service Identifier 2. Discovery Manager 3. Cache System 4. Discovery System and 5. Cloud Model Identifier.

4.3.1 Service identifier

Cloud consumers interact only with the Service Identifier which is a GUI. Service Identifier is responsible for taking the inputs and returning the result back to the CC. Once the CC requests for finding CSPs based on their requirements, Service Identifier takes all the QoS inputs and gives it to the Discovery Manager.

4.3.2 Discovery manager

Discovery manager is responsible to discover all the CSP's that matches the cloud consumers requirements. The Discovery Manager does it in following process.

Check in cache memory Discovery manager first checks if there is any cache available or not for cloud consumers requested QoS attributes using HasQoSCache() method without loading any other modules, it returns true if the QoS caches is available, then it loads and calls the Cache System, once the Cache System is loaded it discoverers available cache for cloud consumers QoS requirements and returns the result back to discovery manager.

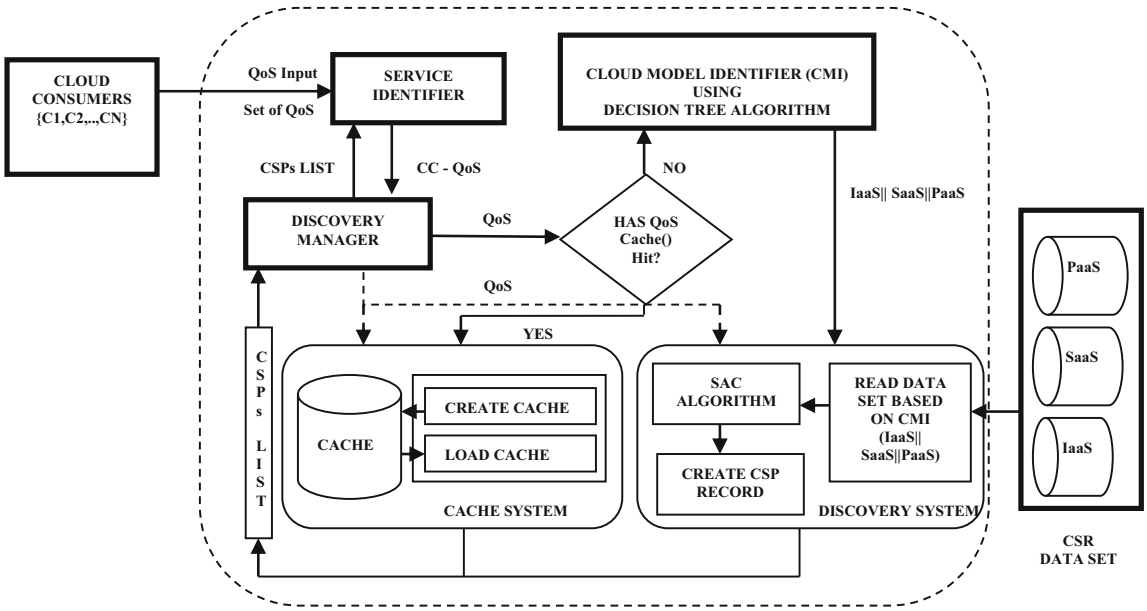


Fig. 7 Cloud Discovery System Architecture

Identify cloud model and discover CSPs HasQoSCache() returns false if the cache is not available, then the Cloud Model Identifier is called to identify the service model for given QoS requirements by the cloud consumer, so that discovery process can be done faster. Once the cloud model type {IaaS, PaaS, SaaS} is identified, the Discovery manager then loads the Discovery system which is responsible for discovering all the CSP's that matches CC's requirement based on the service model identified. The identified cloud model and CC's requirements QoS input is then given to the Discovery system for further discovery of CSPs. After receiving the inputs (QoS, Identified Cloud Model Type) from Discovery manager, Discovery system starts Split and Cache (SAC) Algorithm that discoverers all the matching CSP's. Simultaneously the discovered CSP's list in every iteration is added to cache through Cache System. After completion of SAC Algorithm, it returns the final list of CSP's that matches all the QoS requirements and returns the result back to the Discovery manager.

Algorithm 3 - Split and Cache

```
Input:
    cmt - Identified cloud model type
    cn  - QoS name
    β   - QoS value
    sr  - Split rule to pick CSP
Output:
    af - All founded CSP's
 1. global
 2.     CacheDir = "Caches/"
 3.     DatasetDir = "Dataset/"
 4. end global
 5. function SplitAndCache(cmt, cn, v, sr)
 6.     sn = cmt + cn + v
 7.     if HasCache(sn) then
 8.         Node = LoadCache(sn)
 9.         return Node
10.     end if
11.     ds = LoadDataSet(cmt)
12.     af = fg
13.     for r in ds do
14.         cd = r.Column(cn)
15.         valid = sr.Split(cd, v)
16.         if valid == TRUE then
17.             af.add(r)
18.         end if
19.     end for
20.     CreateCache(sn, af)
21.     return af
22. end function
```

Serve the result to CC Once Discovery Manager receives output i.e., discovered CSP's list from Cache System or from Discovery System, it serves the result to CC through Service Identifier.

The SAC Algorithm 3 consisting of seven major steps:

Step 1. (lines 1–4): Global section before the SplitAndCache() defines all the global variables. The variable CacheDir holds the location for cache directory in server system where the discovery system executes.

Similarly, The DatasetDir tells the location of data sets - {IaaS, PaaS, SaaS}. So, whenever algorithm needs to load data set or need to create caches it uses these two locations of File system.

Step 2. (lines 5–22): Between line 5 to 22 method SplitAndCache() is defined. This method is called multiple time based on the CC's QoS requirement with 4 parameters: 1. cmt - Cloud model type, 2. cn - QoS name 3. v - Value, 4. sr - Split rule. Here sr or Split rule is a Java interface class object, whenever the Split() method of this interface is called, it triggers its definition where conditions are specified based on the QoS and it's data type. Once the execution is over it returns Boolean value true or false which is used to pick CSP's record.

Step 3. (line no. 6): This line concatenates three strings 1.cmt, 2.cn, 3.v and assigns it to variable sn that always creates one unique name for cache file. Later this variable is used in many places to manipulate caches.

Method 1- Has Cache

```
1. function HasCache(sn)
2.     File = CacheDir + sn
3.     if File.isFile( ) and File.exists( ) then
4.         return TRUE
5.     end if
6.     return FALSE
7. end function
```

Step 4. (lines 7–10): In this block cache existence is checked with the help of HasCache() method defined in Method 1, this method checks if cache file is physically exist or not in File System. If cache file is present then it is loaded with the help

of LoadCache() method defined in Method 2. Both the methods HasCache() and LoadCache() requires one parameter to be passed as a cache file name here variable sn is passed which is declared in the line 6. Once caches are found and loaded, it is returned. Otherwise remaining block gets executed.

Method 2 - Load Cache

```
1.  function LoadCache(sn)
2.      File = CacheDir + sn
3.      if File.isFile( ) and F ile.exists( ) then
4.          ar = fg
5.          for r in F ile do
6.              ar.Add(r)
7.          end for
8.          return ar
9.      end if
10.     return NULL
11. end function
```

Step 5. (lines 11–12): In the line 11, Cloud model based data-set is loaded that indicates that caches are not present and data-set need to be loaded into memory. Here method LoadDataSet() defined in Method 3 is called with single parameter cmt (Cloud model type). That loads data-set per cloud model type given in parameter and returns it. Later the loaded data-set is assigned into variable ds. In line 12, an empty set is defined and declared, that holds all founded CSP's record.

Method 3 - Load Data Set

```
1.  function LoadDataSet(cmt)
2.      File = DatasetDir + cmt
3.      if File.isFile( ) and File.exists( ) then
4.          ar = fg
5.          for r in File do
6.              ar.Add(r)
7.          end for
8.          return ar
9.      end if
10.     return NULL
11. end function
```

Step 6. (lines 13–19): In this, foreach loop, all CSP's record is iterated one after another, in every iteration single CSP's record is temporarily assigned to variable r. In line 14, QoS value of CSP r for QoS attribute name cn is taken and assigned to variable cd. In line 15 Split() method of interface object sr is called that triggers its definition where conditions are specified based on the QoS and it's data type, after execution of Split() it returns one Boolean value that indicates if QoS of particular CSP satisfies the requirement. If it returns true, then record of CSP is added to the set container af = {}, otherwise record is ignored.

Method 4 - Create Cache

```
1.  function CreateCache(sn, af)
2.      File = CacheDir + sn
3.      if File.isFile( ) and F ile.exists( ) then
4.          File.Delete( )
5.      end if
6.      for r in af do
7.          File.Append(r)
8.      end for
9.      File.Close( )
10. end function
```

Step 7. (lines 20–22): Once the control comes outside the loop, founded CSP list hold by set container af is added to cache with the help of CreateCache() method defined in Method 4. CreateCache() method takes two arguments first is sn (Name of the cache file) - which is already declared in line number 6, and second is data to be stored where set variable - af is passed. This method first checks and clears any existing cache file with the same name, and then it stores the record of CSPs into cache directory defined in line 2 with file name - sn.

Table 4 Cloud Service Discovey Response Time

CSPs Size	Retrieved CSPs	Response Time (in Sec)	
		Without Cache	With Cache
10,000	98	0.260	0.025
40,000	420	0.390	0.030
70,000	680	0.481	0.039
90,000	850	1.9	0.042
140,000	1423	2.00	0.047

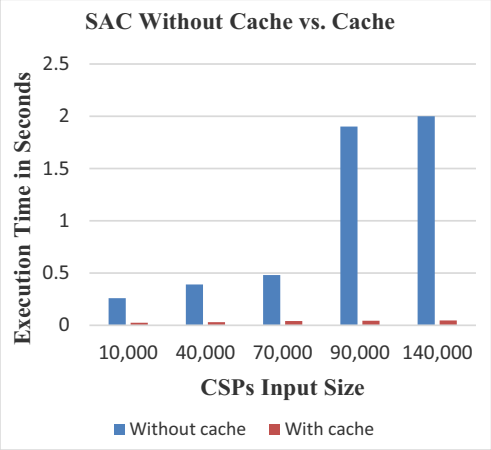


Fig. 8 ResponseTime for Different CSPs Size for 10 dimensions

5 Experiments

5.1 Data description

The platform used for evaluation is Windows 64x, 8 GB primary memory, Intel Core (TM) i5-7200U CPU 2.50 GHz processor and MS SQL server 2012 as database. The algorithms are implemented in CloudSim using JavaFX as front end. The Cloud Service Registry (CSR) allows the CSPs to add, update and delete the cloud services according to the cloud service model. CSR allows CSP's to register their cloud services in two stages. In first stage they provide all the basic details of the cloud service that includes service name, provider name, industry, category, etc. In second stage all the QoS services they are going to advertise with their respective QoS values to the CCs. After CSPs registering the necessary details in CSR, the QoS data is stored into the database according to the cloud model they have chosen.

The Cloud Service Discovery (CSD) allows the CCs to search and discover the cloud services that match their needs. The CSD interface is designed for cloud consumers in such a way that it allows the CCs to pick the cloud service provider according to their application needs. The CSD System starts working based on identification of cloud model type when CCs selects all the required QoS attributes and provide preference values using GUI, based on that CSD will identify the cloud model type and apply SAC algorithm to discover and retrieve the list of CSPs that match the CCs requirements. To

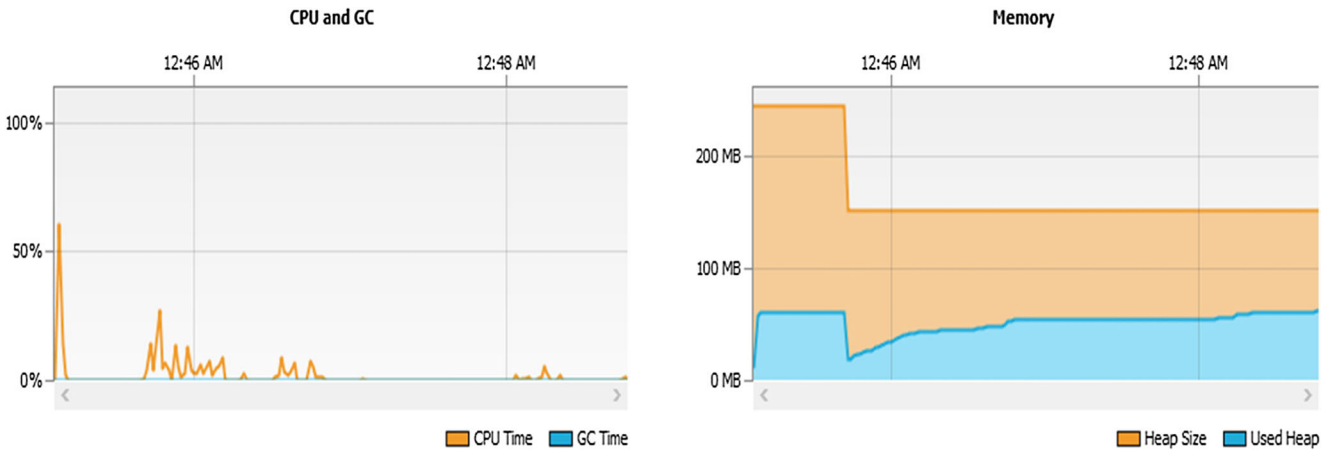


Fig. 9 Cloud Service Registry System Performance Based on CPU, GC and Memory

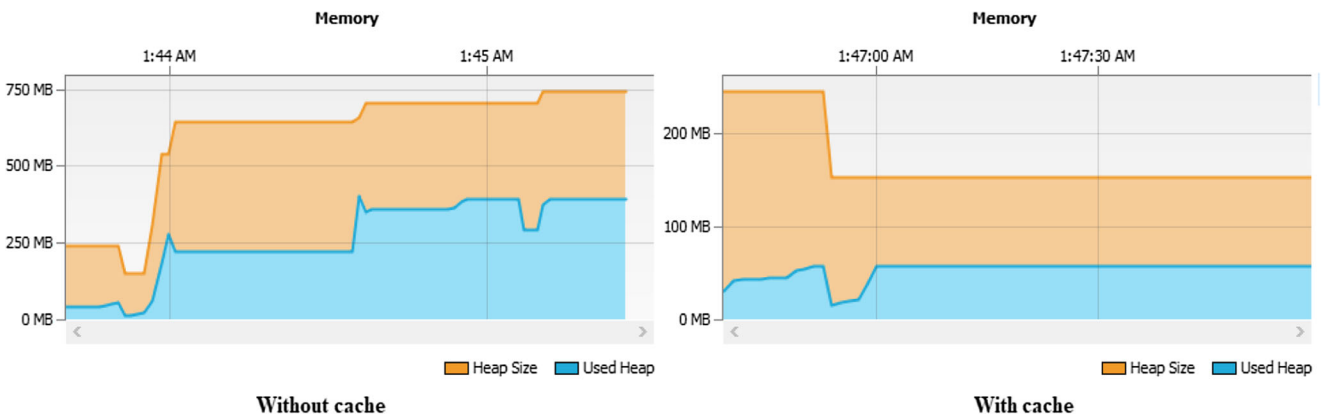


Fig. 10 Cloud Service Discovery System Performance of Memory without Cache and with Cache

Table 5 Response Time of Skyline vs SAC Algorithm

CSPs Size	Skyline Execution Time (in Sec)	SAC Algorithm Execution Time	
		Without cache	With cache
5000	1.5	0.11	0.009
10,000	3	0.22	0.018
30,000	11	0.28	0.024
50,000	24	0.37	0.029

evaluate Cloud service identification and discovery, we perform a large scale synthetic data on 140,000 Cloud service providers for each of the cloud service models with 27 attributes for IaaS, 25 attributes for SaaS and 10 attributes for PaaS. The response time values, CPU consumption and memory utilization where recorded. Response time denotes to CCs sending out a request for CSPs along with their services based on their requirements and getting a response. We generated over 140,000 CSPs with arbitrary values and 10 attributes and number of CSPs size varying from 10,000 to 140,000. We then measured the execution time based on the CCs requirements to retrieve the list of CSPs with cache and without cache of SAC algorithm.

As shown in Table 4, the minimum execution time is 0.368 s without cache and 0.065 s with cache for 10,000 CSPs and the maximum execution time is 2.088 s without cache and 0.165 s with cache for 140,000 CSPs input size and Fig. 8 shows the comparison of execution time between with cache and without cache with varying CSPs input size.

CSR performance shown in Fig. 9 always take static memory and CPU to work, as it never loads any part of the dataset into the memory and it does not requires addition memory and CPU. The CSR Algorithm also takes linear time to store or append the CSP's data into the existing list, even after increasing the size of dataset it does not affects the CSR system in any way. After executing the algorithm and application more than 300 s with three different datasets {IaaS, PaaS, SaaS} that includes 10,000+ records of cloud service providers, the memory consumption never goes out of 250 MB, and also CPU consumption is stable.

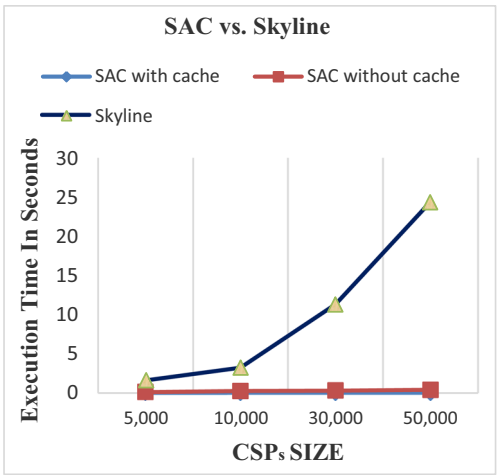


Fig. 11 ResponseTime for Skyline vs SAC Algorithm in Seconds for 6 dimensions

The CSD system has variation in performance as it uses caches to work faster, with caches it does not require to load any dataset into the memory, the internal tree structure is only loaded and treated as a virtual dataset until the caches are cleared or system has to load the original dataset. The CSD system performance without cache and with cache is shown in Fig. 10. The system has loaded around 3500 records into the memory at first run without cache. In the second run same QoS attributes and values are passed but at this time, only the internal tree structure is loaded with cache, which improves the performance of the system, and also shows that performance of the system becomes more stable with a greater number of queries and CCs requests.

5.2 Performance comparison

To study the Cloud service discovery, we compare our work with Skyline [54] and ELECTREIsSkyline algorithm [55]. We did for an input size varying from 5000 to 50,000 CSPs with 6 dimension for Skyline and 10 dimensions for ELECTREIsSkyline algorithm. Table 5 and Fig. 11 shows the response time for Skyline and SAC algorithm with cache and without cache with 6 dimensions. Table 6 and Fig. 12 shows the response time for ELECTREIsSkyline and SAC algorithm with cache and without cache with 10 dimensions. Among all the discovery algorithms SAC approach attains the best response time in all the experimental evaluation constantly. Compared with other discovery approaches, SAC method attains better prediction accuracy, since it takes in to account the CCs preferences while retrieving the list of CSPs.

6 Conclusions

With numerous CSPs providing similar QoS services, finding efficient and relevant QoS services based on CCs requirements is becoming an essential issue in cloud environment.

Table 6 Response Time of ELECTREIsSkyline vs SAC Algorithm

CSPs Size	ELECTREIs Skyline Execution Time (in Sec)	SAC Algorithm Execution time	
		Without cache	With cache
5000	2	0.14	0.012
10,000	1	0.26	0.025
30,000	3	0.33	0.028
50,000	3.7	0.423	0.033

To address this issue, in this paper we have carried out a considerable analysis on cloud service models namely IaaS, SaaS, PaaS service and identified each of the QoS attributes. We have developed a standard cloud service registry system that collects the various QoS service offerings from the CSPs to advertise their services from the identified attributes and have categorized them based on their service model, then produced three datasets that store the information of the categorized service. Based on the QoS information and their values collected from the CCs, an identification algorithm is applied using decision tree on categorized dataset to find the category of the service model the CC wants. Once the service model is identified, Split and Cache (SAC) algorithm is used to discover the various CSPs and their QoS services on the identified dataset that satisfies the CCs demands. Implementation of prototypical results of the proposed technique shows its efficiency and effectiveness with accurate list of CSPs with minimal time complexity and less utilization of CPU and Memory compared with other approaches in both the cases of with cache and without cache.

Our ongoing future research includes integrating direct and indirect Cloud consumers and CSPs feedback while discovering cloud services in accord with service level agreement values and rank the discovered CSPs based on overall and individual QoS attributes and forward the final CSP to the CC based on their QoS preferences.

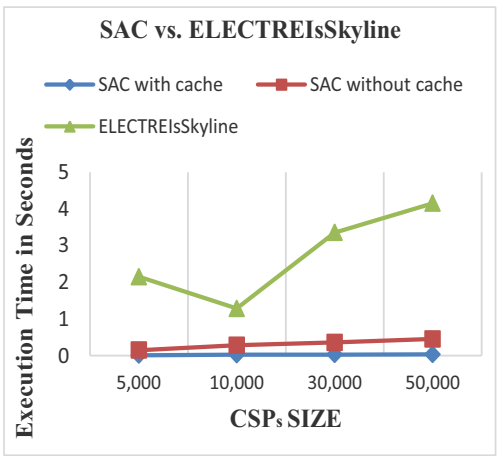


Fig. 12 ResponseTime for ELECTREIsSkyline vs SAC Algorithm in Seconds for 10 diemsnions

References

1. Enslow PH (1978) What is a" distributed" data processing system? Computer 11(1):13–21
2. Casselman S (1993) Virtual computing and the virtual computer. In: FPGAs for Custom Computing Machines, 1993. Proceedings. IEEE Workshop on (pp. 43–48). IEEE
3. Bell M (2008) Service-oriented modeling: service analysis, design, and architecture. John Wiley & Sons
4. Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared. In: *Grid Computing Environments Workshop, 2008. GCE'08*(pp. 1–10). IEEE
5. Roman D, Keller U, Lausen H, De Bruijn J, Lara R, Stollberg M, Fensel D (2005) Web service modeling ontology. Appl Ontol 1(1): 77–106
6. Vaquero LM, Rodero-Merino L, Caceres J, Lindner M (2008) A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review 39(1):50–55
7. Cheng D (2008) PaaS-onomics: A CIO's Guide to using Platform-as-a-Service to Lower Costs of Application Initiatives While Improving the Business Value of IT. Tech. rep., LongJump
8. D'Souza M, Ananthanarayana VS (2013) Cloud Based Service Registry for Location Based Mobile Web Services System. In: *Advanced Computing, Networking and Security (ADCONS), 2013 2nd International Conference on* (pp. 108–111). IEEE
9. Rao S, Rao N, Kusuma Kumari E (2009) Cloud Computing: An Overview. J Theor Appl Inf Technol 9(1)
10. Radack SM (2012) Cloud computing: a review of features, benefits, and risks, and recommendations for secure, efficient implementations
11. Sims K (2009) IBM Blue Cloud initiative advances enterprise cloud computing. URL: http://www-03.ibm.com/press/us/en/pressrelease/26642
12. Schubert L, Jeffery K, Neidecker-Lutz B (2010) The future of cloud computing: Opportunities for European cloud computing beyond 2010. *Expert Group report, public version*, *1*
13. Zhang Q, Cheng L, Boutaba R (2010) Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications 1(1):7–18
14. Thibodeau P (2010) Frustrations with cloud computing mount. *Computer World*
15. Wei Y, Blake MB (2010) Service-oriented computing and cloud computing: Challenges and opportunities. IEEE Internet Comput 14(6):72–75

16. Meshkova E, Riihijärvi J, Petrova M, Mähönen P (2008) A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. Comput Netw 52(11):2097–2128
17. Al-Masri E, Mahmoud QH (2008) Investigating web services on the world wide web. In: Proceedings of the 17th international conference on World Wide Web (pp. 795–804). ACM
18. Mian AN, Baldoni R, Beraldi R (2009) A survey of service discovery protocols in multihop mobile ad hoc networks. IEEE Pervasive Computing 8(1)
19. George K, Kyriazis D, Varvarigou T, Oliveros E, Mandic P (2012) Taxonomy and state of the art of service discovery mechanisms and their relation to the cloud computing stack. In: Grid and Cloud Computing: Concepts, Methodologies, Tools and Applications (pp. 1803–1821). IGI Global
20. Sun Service Registry for SOA (2005) Retrieved from http://xml.coverpages.org/ni2005-06-15-a.html
21. Sim KM (2012) Agent-based cloud computing. IEEE Trans Serv Comput 5(4):564–577
22. Kang J, Sim KM (2011) A cloud portal with a cloud service search engine. In: International Conference on Information and Intelligent Computing IPCSIT (Vol. 18)
23. Resnik P (1999) Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. J Artif Intell Res 11:95–130
24. Kourtesis D, Paraskakis I (2008) Combining SAWSDL, OWL-DL and UDDI for semantically enhanced web service discovery. In: European semantic web conference (pp. 614–628). Springer, Berlin, Heidelberg
25. Kang J, Sim KM (2011) Towards agents and ontology for cloud service discovery. In: Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on (pp. 483–490). IEEE
26. Ranjan R, Zhao L, Wu X, Liu A, Quiroz A, Parashar M (2010) Peer-to-peer cloud provisioning: Service discovery and load-balancing. In: Cloud Computing (pp. 195–217). Springer, London
27. Goscinski A, Brock M (2010) Toward ease of discovery, selection and use of clusters within a cloud. In: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (pp. 289–296). IEEE
28. Zhou J, Abdullah NA, Shi Z (2011) A hybrid P2P approach to service discovery in the cloud. International Journal of Information Technology and Computer Science 3(1):1–9
29. Zeng W, Zhao Y, Zeng J (2009) Cloud service and service selection algorithm research. In: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation (pp. 1045–1048). ACM
30. Zeng C, Guo X, Ou W, Han D (2009) Cloud computing service composition and search based on semantic. In: IEEE International Conference on Cloud Computing (pp. 290–300). Springer, Berlin, Heidelberg
31. Sun L, Dong H, Hussain FK, Hussain OK, Chang E (2014) Cloud service selection: State-of-the-art and future research directions. J Netw Comput Appl 45:134–150
32. Churchman CW, Ackoff RL, Arnoff EL (1957) Introduction to operations research
33. Saaty TL (1996) Decisions with the analytic network process (ANP). University of Pittsburgh (USA), ISAHP, 96
34. Saaty TL (1980) The Analytic Hierarchy Process for Decision in a Complex World. RWS Publications, Pittsburgh
35. Godse M, Mulik S (2009, September) An approach for selecting software-as-a-service (SaaS) product. In: Cloud Computing, 2009. CLOUD'09. IEEE International Conference on. IEEE, pp 155–158
36. Karim R, Ding C, Miri A (2013) An end-to-end QoS mapping approach for cloud service selection. In: Services (SERVICES), 2013 IEEE Ninth World Congress on (pp. 341–348). IEEE
37. Silas S, Rajsingh EB, Ezra K (2012) Efficient service selection middleware using ELECTRE methodology for cloud environments. Inf Technol J 11(7):868
38. Menzel M, Schönherr M, Tai S (2013) (MC2) 2: criteria, requirements and a software prototype for Cloud infrastructure decisions. Software: Practice and experience 43(11):1283–1297
39. Limam N, Boutaba R (2010) Assessing software service quality and trustworthiness at selection time. IEEE Trans Softw Eng 36(4):559–574
40. Li A, Yang X, Kandula S, Zhang M (2010) CloudCmp: comparing public cloud providers. The 10th annual conference on Internet measurement. ACM, New York, pp. 1–14
41. Rehman Z, Hussain OK, Hussain FK (2012) IAAS cloud selection using MCDM methods. In: 2012 IEEE Ninth international conference on e-business engineering (pp. 246–251). IEEE
42. Jeong HY (2013) The QoS-based MCDM system for SaaS ERP applications with Social Network. J Supercomput 66(2):614–632
43. Kanagasabai R, Ngan LD (2012) Owl-s based semantic cloud service broker. In: Web Services (ICWS), 2012 IEEE 19th International Conference on (pp. 560–567). IEEE
44. Garg SK, Versteeg S, Buyya R (2013) A framework for ranking of cloud computing services. Futur Gener Comput Syst 29(4):1012–1023
45. Cloud Service Measurement Index Consortium (CSMIC), SMI framework. URL: http://beta-www.cloudcommons.com/servicemeasurementindex
46. Garg SK, Versteeg S, Buyya R (2011) Smicloud: A framework for comparing and ranking cloud services. In Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on (pp. 210–218). IEEE
47. Sundareswaran S, Squicciarini A, Lin D (2012) A brokerage-based approach for cloud service selection. In: Cloud computing (cloud), 2012 IEEE 5th international conference on (pp. 558–565). IEEE
48. Tran VX, Tsuji H, Masuda R (2009) A new QoS ontology and its QoS-based ranking algorithm for Web services. Simul Model Pract Theory 17(8):1378–1398
49. Afify YM, Moawad IF, Badr NL, Tolba MF (2013) A semantic-based software-as-a-service (saas) discovery and selection system. In: Computer Engineering & Systems (ICCES), 2013 8th International Conference on (pp. 57–63). IEEE
50. Goud S (2016) Software Metrics for SAAS, PAAS, IAAS-A Review. International Journal for Research in Applied Science & Engineering Technology, Volume 4 Issue 5, IJRASET
51. Wu CS, Khoury I (2012) Tree-based search algorithm for web service composition in SaaS. In Information Technology: New Generations (ITNG), 2012 Ninth International Conference on (pp. 132–138). IEEE
52. Kullback S, Leibler RA (1951) On information and sufficiency. Ann Math Stat 22(1):79–86
53. Sharma H, Kumar S (2016) A survey on decision tree algorithms of classification in data mining. International Journal of Science and Research (IJSR) 5(4):2094–2097
54. Idrissi A, Abourezq M (2014) Skyline in cloud computing. Journal of Theoretical & Applied Information Technology 60(3)
55. Abourezq M, Idrissi A (2015) Integration of QoS aspects in the cloud computing research and selection system. arxiv preprint arxiv:1702.04966