Front. Comput. Sci., 2019, 13(3): 599–617
https://doi.org/10.1007/s11704-017-6124-7

REVIEW ARTICLE

Cloud service selection using cloud service brokers: approaches and challenges

Meysam VAKILI (✉)[1], Neda JAHANGIRI[1], Mohsen SHARIFI[2]

1 Department of Computer Engineering, University of Science and Culture, Tehran 14619-68151, Iran
2 School of Computer Engineering, Iran University of Science and Technology, Tehran 16846-13114, Iran

Abstract Cloud computing users are faced with a wide variety of services to choose from. Consequently, a number of cloud service brokers (CSBs) have emerged to help users in their service selection process. This paper reviews the recent approaches that have been introduced and used for cloud service brokerage and discusses their challenges accordingly. We propose a set of attributes for a CSB to be considered effective. Different CSBs' approaches are classified as either single service or multiple service models. The CSBs are then assessed, analyzed, and compared with respect to the proposed set of attributes. Based on our studies, CSBs with multiple service models that support more of the proposed effective CSB attributes have wider application in cloud computing environments.

Keywords cloud service broker (CSB), cloud service selection, cloud computing, quality of service (QoS)

1 Introduction

The National Institute of Standards and Technology (NIST) defines cloud computing as: "*a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction*" [1]. Cloud service providers have tried to provide as many services as they canto align with this credible definition, resulting in the provision of a variety of services for cloud users. To help and guide users of cloud services to reach their desired services, out of existing provisioned cloud services, a variety of Cloud Service Brokers (CSB) [2–4] have emerged. A cloud service customer's choice of CSB can be critical to the effectiveness of cloud computing services in customers' applications. In a nutshell, our review of CSB approaches aims to facilitate customers' choice of CBS by a cloud service customer.

A CSB can be defined as a service that acts on behalf of a client to provide resources and deploy application components [5–7]. It is an entity that manages the usage, performance, and delivery of cloud services and acts as a mediator for negotiation between cloud providers and consumers [8].

Grozev and Buyya [9] divided CSB responsibilities into three classes:

1) Automatic resource provisioning and management across multiple clouds.
2) Automatic deployment of application components in the provisioned resources.
3) Scheduling and load balancing of incoming requests to allocated resources.

CSBs have additionally been classified according to their roles and the mechanisms they deploy to perform their brokerage responsibility [2,9,10].

Wadhwa et al. [2] have considered four different roles of CSBs: aggregator, integrator, governance service, and customizer. Sanchez et al. [10] defined three classes of CSB: business cloud brokers, financial cloud brokers, and technical

Received February 29, 2016; accepted April 26, 2017

E-mail: m.vakili@usc.ac.ir

cloud brokers. Grozev and Buyya [9] categorized CSB mechanisms as directly managed and externally managed and further divided the externally managed CSBs as either service level agreement based or trigger-action.

Liu et al. [8] presented one of the most notable categorizations of CSBs as the following three classes: intermediation service, aggregation, and arbitrage. In the intermediation service, a cloud broker upgrades a specific service by improving some definite capabilities and providing additional services for consumers. A cloud broker offering the aggregation service combines some services and integrates them into fewer services or even just one service. In the arbitrage service, a broker integrates unfixed services and grants permission for selecting from few providers.

Although different cloud providers may have designed a different architecture for their brokerage services [11–13], they have all envisaged the same role of cloud service discovery and selection for their CSBs [14]. Cloud service discovery is the process of finding a cloud provider who can best satisfy a consumer's needs, and service selection is a method for requesting discovered services. Every CSB uses its own algorithms and mechanisms for selecting an appropriate service from a huge collection presented by cloud providers in order to satisfy a cloud consumer's needs.

Various studies have been carried out on cloud service selection. Whaiduzzaman et al. [15] reviewed and classified cloud services' provisioning strategies using different criteria. Sun et al. [16] investigated cloud service selection approaches from five different perspectives: decision making techniques, data representation models, parameters and characteristics of cloud services, contexts, and purposes. Subha and Banu [17] surveyed various studies on quality of service (QoS) ranking of cloud computing and cloud service selection. Jula et al. [18] also surveyed cloud service selection approaches and determined nine classes of cloud service composition research approaches.

Despite the existence of a rich repertoire of surveys on cloud computing services and their brokerage, a deep and comprehensive review on the role of CSBs in cloud service selection is still missing.

This paper investigates the approaches and mechanisms of brokerage in cloud computing environments for the selection of cloud services that conform with users' needs.A cloud service is defined as "*any resource that is provided for the users over the Internet*", such as infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). Prominent works with unique ideas are investigated and analyzed with regard to important criteria in the cloud service selection process. Two categories of CSBs are reviewed: CSBs that use the services of just one cloud service provider (called single service models) and CSBs that use the services of more than one cloud service provider (called multiple services models). The main challenges of each work are discussed, summarized, compared, assessed,and reported in tabular form.

The remainder of this paper is organized as follows. Section 2 discusses the brokerage process in cloud computing environments and proposes a set of attributes for determining the effectiveness of a CSB. Section 3 reviews a categorized presentation of eminent CSBs, focusing on their approaches towards service selection. Section 4 presents the most important specific and common challenges in each work. Section 5 compares and evaluates the reviewed CSBs with respect to our proposed attributes, and in Section 6 we give concluding remarks on our paper.

2 Attributes of an effective CSB

We require a set of criteria in order to categorically compare and assess different approaches towards the brokerage of cloud services. Regardless of differences in existing CSB architectures and their architectural components,which are designed according to cloud providers' policies and consumer needs, a generic set of criteria can be envisaged as a set of attributes of an effective CSB. Based on the reviews of studies reported in this paper, we propose the following eight desired attributes for any typical CSB, whose main role is to facilitate service discovery and selection and effectively satisfy the requirements of its cloud customers:

1) Support for various QoS attributes QoS refers to a collection of service qualities or attributes, such as availability, security, performance, scalability, throughput, latency, usability, reputation, and reliability [3,16]. An effective CSB must consider a broad spectrum of both qualitative or quantitative QoS attributes in selecting suitable services. It must also allow cloud customers to select their own QoS attributes in an appropriate way.

2) SLA violation recognition An SLA is a legal contract between providers and consumers,which defines the quality of service and is achieved through a negotiation process [19].

The majority of existing commercial and research-based CSBs use SLA in order to progress negotiations between providers and customers, but very few also include a flexible and suitable mechanism against SLA violation. When an SLA is established, it may display incompetency due to

changes in components, workloads, or external conditions or software or hardware faults [20]. In such circumstances, an effective CSB must identify SLA violations, take proper actions to end such violations, and compensate any accrued losses.

3) Price optimization One vital parameter in selecting a suitable service provider is the cost [21]. Irrespective of the type of cost model the provider uses, CSBs must consider the cost as an important (or even the most important) criterion in service selection. Most CSBs offer the same costs for the same services, but what makes a CSB eminent among others is a mechanism for optimizing customer value for money. Particularly when a CSB supports service composition, the ability of optimizing the cost of those forced-to-compose services is really important.

4) Selection of suitable providers for different types of cloud services By one renowned classification, cloud services fall into one of three classes: IaaS, PaaS, or SaaS. But more recently, all Internet-oriented services have very quickly moved into or towards the cloud, which a growing concern for the everything as a service (XaaS) discipline [22]. Cloud services have a wide spectrum of end users, companies, and organizations with different requirements, so an effective CSB must recognize all types of cloud providers and search among all available cloud services.

5) Customer profiling Regardless of the unique attributes and characteristics of different cloud environments, cloud service selection is a kind of search problem. In every search algorithm, the speed of search is a key factor in evaluating its performance, and many factors play a role in improving search speed. The use of a profile or history of all past requested services of each user can accelerate the search process [3,23].

6) Dynamic brokerage CSBs are faced with cloud users whose requests for cloud services vary dynamically in size and characteristics, while cloud providers' statuses of service provisioning vary dynamically too [24]. The cost of services for providers and the price of services for users also varies dynamically with respect to current supply and demand for resources [4]. Effective CSBs should be equipped with proper mechanisms to dynamically broker and select suitable services to offer to users.They should detect newly registered services and even switch users to new services that could serve users better than their previous one. Thus, CSBs must search periodically for available services.

7) Partial information request on specification of services from customers Most cloud service selection mechanisms assume that cloud customers are competent in using the cloud and can be asked to provide full specification of their requested services [25]. This blanket assumption deprives less experienced and novice customers from using cloud services [16]. It is desirable that CSBs can mediate between cloud providers and less professional cloud customers too, asking for the least information on required services from the customers and figuring out other necessary information about the customers' requested services by other means, such as using user profiles, history files, and recommendation systems. Indeed, CSBs can still cater for professional customers differently and receive full information on services from them.

8) Customer preference prioritization Customers should be able to pass their preferences on functional and non-functional attributes of their requested services on to CSBs. Preferences on non-functional attributes greatly influence the effectiveness of cloud services for customers [26]. The quality requirements can be multi-objective and service consumers can give different priorities to different quality attributes aligned with their quality requirements [27]. An effective CSB must consider the priorities of each customer in the selection of appropriate services from service providers.

3 Cloud service brokers

Having proposed a minimal set of desired attributes for effective CSBs in the previous section, eminent CSBs over the last five years are reviewed in this section; their evaluation with respect to the proposed attributes is presented later in Section 5. Two classes of CSB are differentiated in our review: single service models and multiple service models. Table 1 shows the names of the CSBs that are reviewed in this section.

Acronyms are used for naming different CSBs in this paper, and as such, their names may differ from the title of papers in which they have been reported; for the papers they are reported in, please follow the references given alongside them.

3.1 Single service model CSBs

This subsection investigates the single service model CSBs. These CSBs almost support the IaaS model, but any other CSB designed for a special type of cloud service will also be in this class.

3.1.1 QBROKAGE

QBROKAGE [28] uses a genetic algorithm with four goals: responding to heterogeneous QoS requirements of applications,preventing vendor lock-in, supporting various price

Table 1 Classification of reviewed CSBs

Single service model	Multiple service model
QBROKAGE [28]	AHP & TOPSIS [21]
SLA-based SaaS Provisioning [3]	CSP-index [33]
Brokering Using Game Theory [23]	Two Layered Brokerage [34]
STRATOS [29]	SMICloud [35]
Distributed Cloud Brokering [30]	Brokering for Optimized Placement of Virtual Machines [36]
Multi-Cloud Brokering [24]	
Dynamic Brokering on Federated Clouds [31]	OWL-S Based Semantic CSB [14]
T-Broker [32]	Price Optimization using 0-1 Knapsack [4]

models, and appropriate scaling while preserving interactivity.

In QBROKAGE, requested user services (called appliances), resources, and QoS attributes are modeled (Anastasi et al. [37]). An appliance is represented by a non-directional graph, wherein each vertex displays an appliance and each edge displays a connection path. Each provider is also modeled as a data center containing a collection of hosts running a number of virtual machines (VMs). QoS attributes are classified as ascending, descending,or equal [38]. These classes correspond to the amount of user demand for each QoS attribute and the real amount that a provider can supply for each. To cater for price model differences used by various providers, two models of pre-resource and pre-VM price are supported.

Figure 1 shows the overall architecture of QBROKAGE. The customer sends their request application (service specs) in the standard OVF (open virtualization format) to QBROKAGE. The OVF parser component analyzes the OVF input and creates a graph application. The application is added to the *application queue*, which buffers current requests to map an application and manages the delivery to the *mapping* component. Given the datacenter and application properties, the mapping component computes a set of application mappings. Other required information for the mapping component such as the characteristics of the network connecting data centers and provider conditions are supplied by the *monitoring* and *net estimator* components. When one or more map plans are computed, they are sent to the *allocator* component as input. Eventually, the allocator component assigns VMs for each mapping plan in turn.

The mapping component is the main part of QBROKAGE; it uses a genetic algorithm to map applications to VMs and data centers. The genetic algorithm allows constraints to be added with the least interference. The represented brokerage algorithm uses the main procedure of the standard genetic algorithm. The difference is that each solution (chromosome) is a vector that represents the allocation mapping of each appliance/VM to providers, and each cell represents one application. The fitness function is also computed based on relationshipss between QoS supplied by cloud providers and QoS requested by users.

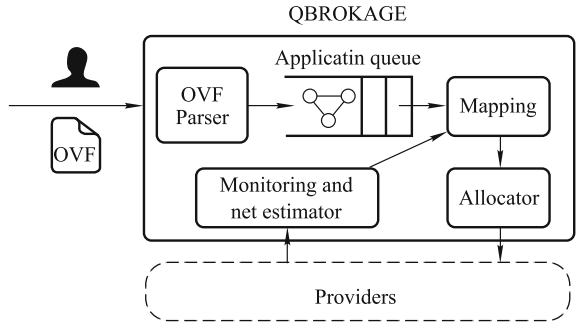


Fig. 1 QBROKAGE architecture [28]

3.1.2 SLA-based SaaS provisioning

Badidi [3] proposed a framework in which a broker is in charge of selecting an appropriate service provider in accordance with QoS requirements of the service consumer and negotiation of SLA conditions. As shown in Fig. 2, the proposed framework is divided into four parts: consumer, broker, monitoring infrastructure, and service providers. The broker has managing functions,including access controls, management policies, SLAs, and service supplements. All parts are monitored by the coordinator (broker). The broker's back-end database holds managing policies, SLAs, and QoS information. The selection manager applies the function of provider management and it is in charge of applying different policies to select an appropriate provider based on consumer considered QoS parameters. The monitoring infrastructure consists of a monitoring adapter and measurement services. There is a monitoring adaptor for each measurement service that maps QoS metrics to SLA parameters, assesses the level of recent services, and evaluates an agreement with the SLA.

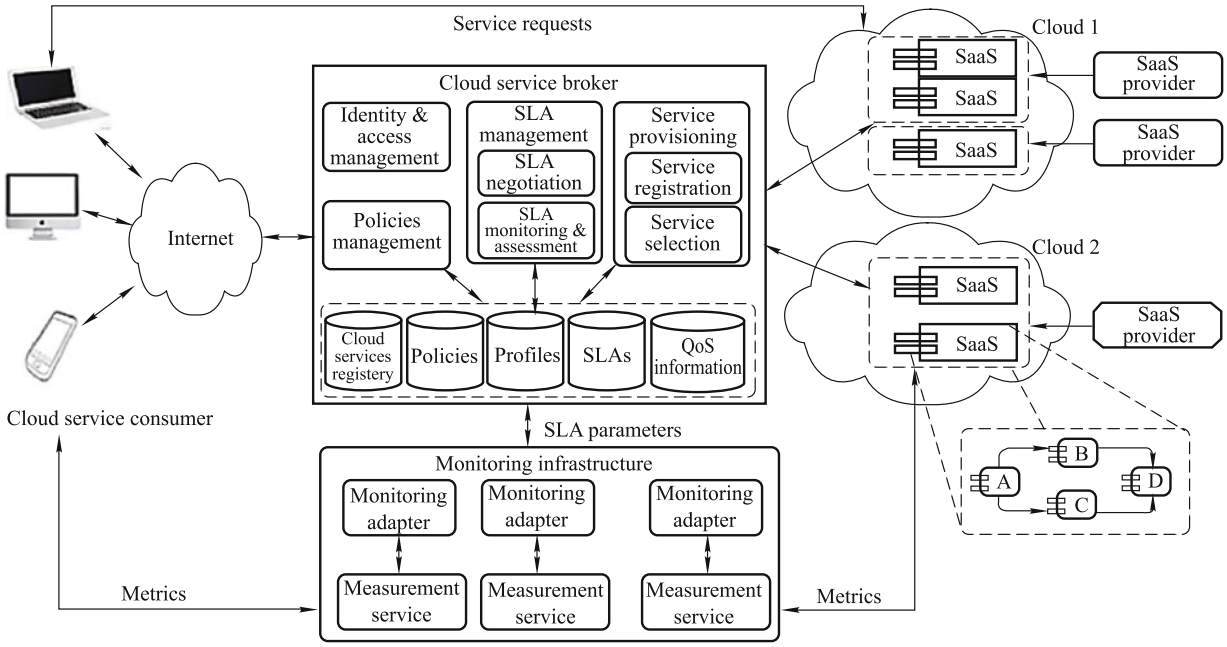


Fig. 2 SLA-based SaaS provisioning framework [3]

The negotiation process starts with the consumer SLA request and the result is sent to the broker. After authenticating the consumer, the broker requests the customer profile from the *profile manager*. Then a request for selecting an appropriate provider and receiving service supplier policies are sent to the selection manager and policy manager, respectively. If the service consumer specification exists in the profile repository, the coordinator can recognize whether the proposed provider can provide the considered service. If there is no profile for the consumer, the coordinator asks the consumer for information about assumed quality level in order to complete the profile. If at least one provider can supply the user's needs, the SLA manager sends the coordinator's request to the user. If the SLA request is acceptable to the provider, the coordinator analyzes it to check whether it satisfies the user's functional or non-functional needs.

Having found an appropriate SLA, the certificate is sent to the provider. After receiving the SLA certificate from the broker, the SLA manager registers the consumer as a member and the SLA is ready to be applied.

3.1.3 Brokering using game theory

Ray et al. [23] presented a broker that uses the game theory model for automatic SLA negotiations to provide an optimized amount of price and quality for both provider and customer. Figure 3 shows the broker's architecture. The main modules of the broker are as follows:

- Resource request requirement This module saves the details of user requests. A user request includes the name of the task to be computed, the task type, and an SLA template. This request is sent to the request analyzer module.
- Request analyzer A request is first analyzed by looking at the history. If details of the task exist in the history, proper measures are adopted. Otherwise, task details and task template are sent to the service provider broker and SLA service module, respectively.
- History This module holds the completion times of requested tasks and their related SLA documents. In the case that a new requested task exists in the history, a proper resource is selected faster.Otherwise, a new entry is created for the task in the history.
- Service provider broker This module assumes the duty of administrating a collection of IaaS providers. The service provider broker acts as the owner of virtual resources that some service providers have registered on them as member. The service provider broker also records details of registered providers and their corresponding SLA details.
- Execution time analyzer This module estimates the task completion time statically. It also computes the execution duration of the task and sends the result to the request analyzer module.
- SLA service This module takes two input parameters

of task price and the quality of selected instance and starts the SLA negotiation process using game theory. In fact, the SLA service module receives SLA templates related to a provider and a user and at each round of the game tries to decrease the difference in satisfaction level at the equilibrium point. Optimized value is gained when the difference in satisfaction level between user and provider at equilibrium level is close to or equal to zero. Finally, an optimized value for price and quality is calculated and sent to the resource selection module for service selection.

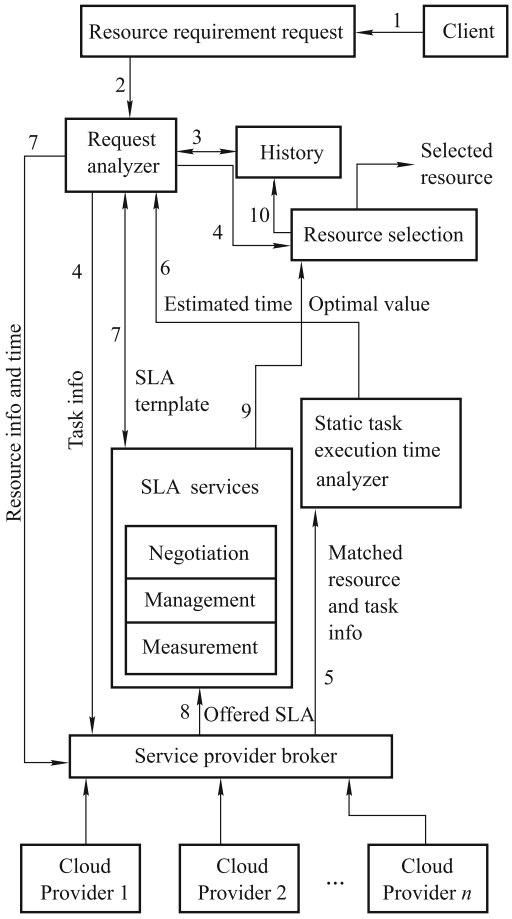


Fig. 3 Brokering using Game Theory [23]

3.1.4 STRATOS

This brokering method [29] provides resources in inter-cloud. STRATOS lets application developers identify their needs in terms of key performance indicators. When a service request is sent to the broker, the request is sent to all providers and the best resource is selected. This application distribution decreases topology price and satisfies user goals better. The problem of resource acquisition decision (RAD) has been formulated as a multi-criterion optimizing problem [39]. Each RAD problem consists of selecting N resources from M limited providers.

According to Fig. 4, a topology descriptor file (TDF) is defined by the user to describe the application topology the user wants to be deployed in the cloud. The user appoints many application requirements in this document, such as required clusters, servers, web hosts, container, and broker inputs. After receiving the topology document, the cloud manager contacts the broker to create a topology instance. The broker does primary RAD computing (resource allocation in both providers). Configuration and its attributes are defined in a third application programming interface (API) layer [40].

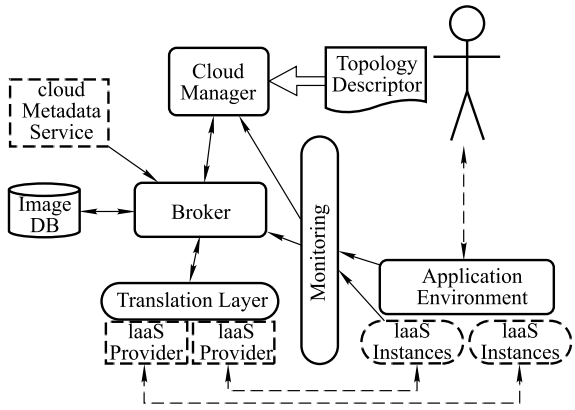


Fig. 4 STRATOS architecture [29]

More preferred user limitations and commitments are defined in a goal function. STRATOS focuses on the two goals of price and lock-in prevention. The broker needs information on two classes of developers to solve the RAD problem: configuration and a collection of goals. Configuration is defined by a list of attributes, each one consisting of a triple (name, value, and unit). Each goal indicates a computed utility function for topology. The broker tries to optimize the goal set that is defined in the TDF. The weight of the goal collection is used as default, but optimizing method can be defined by the user.

Selection process takes place in two stages: possible configuration identification and goal optimization. The broker selects a set of goals that satisfy defined goals in the TDF. In the second stage, a set of equivalent configurations are selected by a multi-criterion optimizing process, resulting in an appropriate provider. When no configuration satisfies the goals, the broker attempts to find the closest configuration for each property.

3.1.5 Distributed cloud brokering

This is a distributed multi-user brokering method that selects

resources from different providers dynamically. The broker uses a document called CFP (call for proposal), which includes the customer's needs and policies, such as price in time unit and availability. CFP includes two documents: SLA describing model in the form of XML SLA@SOI schema and exchanging policy that consists of a set of rules to be enforced in service selection by the broker. To get the SLA description model from the user in the CFP document, a graphic interface has been designed that takes four pieces of information from the user about configuration of user essential resources, technical requirements of applications, service level, and service conditions. Alongside this information, the user enters an SLA description template and soft/hard constraints on browser policy as inputs so that the objective function can be optimized.

Users and vendors, who have been identified by usernames and passwords, send their CFP proposals in parallel and asynchronously to the system and receive acknowledgements. While their requests are pending, users can wait for broker results. When one of the brokers finds a feasible proposal for that request, current results are updated. The exchange ends when there is no special proposal for optimizing the user query.

The broker uses the databases twice, once for user management, such as for checking their profiles and credentials therein, and once for indexing proposals and querying the SLA candidate. Agents withdraw different proposals from the queue and index them to allow different brokers to query the database and retrieve the SLA candidate to be brokered.

3.1.6 Multi-cloud brokering

The multi-cloud brokering CSB [24] has two main components. The cloud broker administrator configures the broker before use. Configuration involves defining a provider list containing their information and an instance list consisting of instance type and its prices in each cloud. The second component is the cloud broker user and is responsible for identifying a new service by a service description file upon receiving cloud information and the instance list. Service description is a user defined file containing accurate information on the service to be deployed by the broker: information such as service components, optimization criteria (such as cost, performance, resource consumption, energy consumption, or a combination of all), scheduling parameters and policies, instance types, and instance efficiency.

The broker has a central database that saves different lists that have been used by other components. The cloud list saves provider information. The instance list saves information about instance types in each cloud as well as their prices and pricing models. The service list saves information about services identified by the user. Each service is described in one service description file. The VM list saves information about the VMs managed by the broker in different clouds. The list has each VM's current status and information about its scheduling and resource consumption.

The broker has three main components, namely the scheduler, the VM manager, and the cloud manager:

- The scheduler reads the service list and uses the service description file as an input for deploying new services. It recalls the schedule module defined in the service description, then specifies whether the VM list must be deployed or cancelled in each cloud. It then updates the VM list to notify the VM manager. Before each scheduling action, the scheduler reads the instance list to become aware of instance types, prices, and the number of available instances in each cloud.
- The VM manager reads the VM list periodically and calls appropriate functions regarding the VM's current conditions. It is also in charge of monitoring the resource value for each VM and updating the VM list.
- The cloud manager collects information and prices from available instances periodically for all instances in the instant list and updates the list. This is done to allow dynamic pricing.

3.1.7 Dynamic brokering in federated clouds

Resource allocation in the cloud can be done by multi-agent systems [41]. Dynamic brokering on federated clouds [31] is an exemplar CSB that uses three agents: consumer agent (CA), resource brokering agent (RBA), and resource provider agent (RPA).

Resource allocation is straightforward when requested resources are available in one cloud provider, but more complicated if requested resources are available in a collection of cloud providers (e.g., in federated clouds). In the latter case, the resource allocator must be informed about the providers' services and the status of each provider in the federated clouds. Each resource may have different prices in different cloud providers; prices follow a supply and demand model. Therefore, acquiring the current price of resources from different providers is very difficult. Resource allocation is based on received feedbacks from users. The broker agent checks each provider in turn by negotiating with the

provider's agent. If the consumer's need is not completely satisfied by the provider, the broker starts negotiating with another provider.

Agents use three algorithms for negotiation: CA communication, RBA communication, and RPA communication. Figure 5 shows a flow diagram of the negotiation process. CA creates a request as a CFP message and sends it to RBA. RBA extracts resources and starts searching for best quality and lowest price for each resource. Using the resource list, it computes the total price of resources and sends the computed price to CA. If the price is acceptable to CA and RBA receives an *ACCEPT* message from CA, it creates a CFP message and sends it to RPA. RPA extracts resources and their total price. If total price is equal to or less than the price specified in the CFP, the request is accepted. Then RBA updates the provider list with this demand price ratio and repeats the same procedure. If the agreement is acceptable to CA, it sends an AGREE message to RBA, which in turn sends a *CONFIRM* message to RPA. Otherwise, CA sends a *REFUSE* message and the protocol starts all over again.

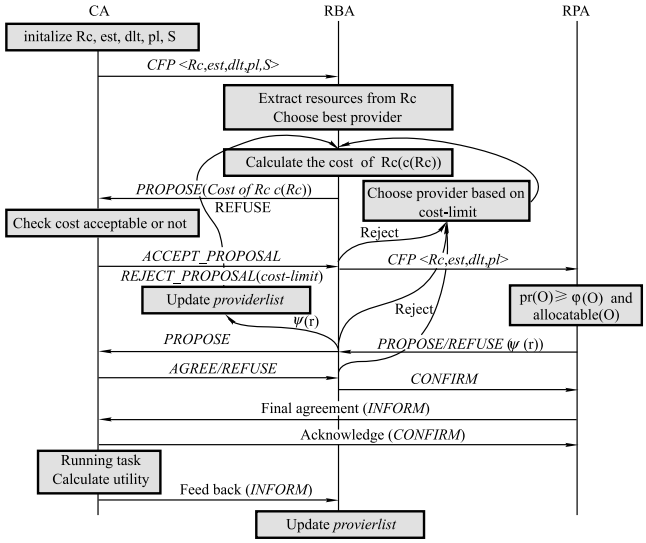


Fig. 5 Data flow diagram of dynamic brokering on federated clouds [31]

When PRA receives a CONFIRM message, the final agreement is defined by sending a CONFIRM message to CA, and CA displays *INFORM* for acknowledgement. After finishing the task, the consumer computes resource efficiency and sends its feedback to RBA. RBA updates the provider list with received values. In this way, consumers get their requested resources with the lowest prices without needing to know the locations or the prices of resources.

3.1.8 T-Broker

Trust management is a complicated and difficult task in distributed environments whose resources are geographically distributed and belong to distinct organizations [42], such as multi-cloud environments. T-Broker [32] is a trust-aware CSB, which acts as a trusted third party for trust management and resource scheduling in multi-cloud environments. It uses a hybrid trust mechanism, which is based on a combination of feedback-based trust and the trust based on real-time and multi-source service data. Furthermore, it uses a maximizing deviation method to compute the trust of a service resource, in which the trusted attributes are weighted manually or subjectively.

Figure 6 shows the following five modules of the T-Broker:

- Sensor-based service monitoring This module monitors service parameters dynamically and is responsible for getting run-time service data of allocated resources. To calculate the trustworthiness of a resource [39], T-Broker uses two types of software sensors: monitoring sensors for collecting the performance information of computing resources, such as CPU frequency, memory size, hard disk capacity, and network bandwidth and computing sensors for collecting and computing QoS information, such as the current CPU utilization rate, current memory utilization rate, current hard disk utilization rate, current bandwidth utilization rate, average response time, and average task success ratio.
- Virtual infrastructure manager (VIM) This module is used to collect and index all resource information from cloud providers and to act as a resource management interface to a monitoring system. VIM, which is based on OpenNebula, is also responsible for the deployment of each VM in the selected cloud and the management of the life-cycles of VMs.
- SLA manager and trusted resource matching This module selects and composites highly trusted resources for users from a trusted resource pool according to the SLA contract.
- Hybrid and adaptive trust computation model This module, which is the core of the T-Broker, uses a hybrid and adaptive trust model to compute the overall trust degree of service resources. Trust is defined as a fusion evaluation result by adaptively combining real-time service behavior with the social feedback of the service resources. This module allows users to specify their requirements and preferences to get a customized trust value of the cloud providers.
- Services feedback and aggregation T-Broker uses

a lightweight feedback mechanism among virtualized data centers and users in order to reduce networking risk and improve system efficiency. This mechanism collects users' ratings and aggregates them to yield a global evaluation score. A user's rating is used as a reference by other users in future transactions.

3.2 Multiple service model CSBs

In addition to single service model CSBs, there are some CSBs that support more than one cloud service model. These CSBs are multi-objective and either work with different cloud services from the beginning or are designed in such a way to have the capability and flexibility to be in agreement with different providers and cloud services models.

3.2.1 AHP & TOPSIS

AHP [21] uses a mechanism based on TOPSIS (technique for order preference by similarity to ideal solution) [43] to select the most appropriate cloud provider. It measures the quality of each cloud provider and prioritizes all providers with respect to customer needs. It considers three roles: requester (which could be an SaaS provider, a PaaS provider, or an end user), provider, and broker.

Selection of an appropriate cloud provider from many cloud providers is a multi-criterion decision-making (MCDM) problem that is solved in three phases:

- Identifying suitable criteria In this phase, services initiative measurement (SMI) is used to define suitable criteria for selecting a cloud provider. SMI is a standard way of measuring cloud services provided from user requests. SMI is developed by a consortium called Cloud Services Measurement Initiative Consortium (CSMIC) [44]. SMI has a hierarchy structure and consists of various attributes, such as accountability, cost, agility, performance, assurance, security, privacy, and usability.
- Evaluating the weight of criteria using APH In this phase,the relative weights of chosen criteria are determined by using each criterion of AHP [45] as an MCDM.
- Ranking cloud providers using TOPSIS In this phase, TOPSIS is used to select an alternative that is closest to the ideal solution and farthest from the negative ideal solution. TOPSIS takes a list of N providers, M related attributes of each provider, and the weight of each computed attribute. It then creates a matrix whose elements are the values of each attribute for each cloud provider. This value is defined by provider attributes. Eventually, using these values and relative computations, a parameter called *relative closeness* is computed for each provider. Each cloud provider whose value is more than relative closeness is the most appropriate alternative according to user needs.

3.2.2 CSP-index

This method [33] uses a unique indexing technique to manage provider information and implements an algorithm to classify providers and aggregate them. The architecture of this broker has two main aspects:

- Indexing cloud service providers This paper uses a method called CSP-index (cloud service provider) to make index keys for providers. The CSP-index has been based on an encoding technique that uses the similarities among various providers. Since providers with the same attributes are saved alongside each other, this method can accelerate the query process. When a bro-

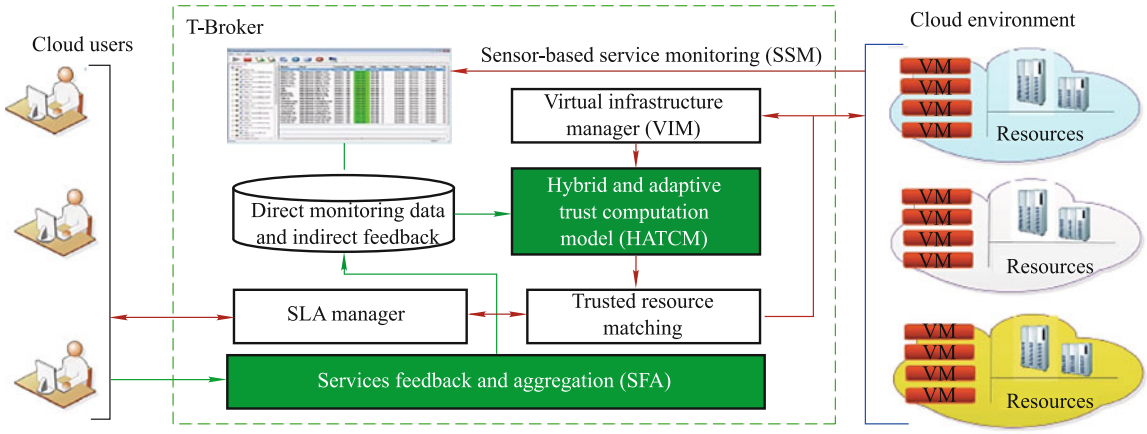


Fig. 6 T-Broker's architecture [32]

ker identifies a candidate provider in the index, it can find other candidate providers with the same attributes immediately. In the CSP-index data structure, inner nodes have a B^+-tree-like format and are used as a search directory. The algorithm used in producing keys and making the CSP-index has three stages: *property encoding* for encoding attributes of each provider, *relationship encoding* for encoding relationship between providers made by subcontracting, and *index key generation* for producing final index keys. Meanwhile, the k-means algorithm and iDistance [46] are used for clustering providers and producing index key,respectively.

- Cloud service selection Indexing helps the broker to select a suitable cloud service. User sends a cloud service selection query (CSS query) to the broker to set the provider's considered values and attributes. The CSS query algorithm is then run to perform four phases: (1) *query encoding*, which transforms user query to a CSP-index form, (2) *K-nearest neighbor search*, which returns *K* candidate service providers whose index keys are similar to the query encoding and hence may satisfy the query requirements, (3) *refinement*, and (4) *consideration of special criteria*. Phases 3 and 4 investigate candidate provider properties and their relationships to find the best combination of providers to meet the user's needs.

3.2.3 Two Layered Brokerage

This approach [34] makes brokers capable of staying in two different layers in the cloud computing architecture and communicating to each other. In fact, we have two kinds of brokers: SaaS broker that selects an SaaS provider, and the cloud broker, which selects a cloud provider. Communication between brokers adds to each broker's information and raises its QoS level, because the QoS level provided by an SaaS provider is mostly dependent on its foundation, i.e., the cloud provider. The aim of this approach is to make suitable communication between these two brokers and select the best service provider for a user.

Two layered brokerage uses a multiple cloud architecture (Fig. 7) with five components [47,48]. The user sends their request containing functional and non-functional attributes to the SaaS broker. When the broker finds a suitable SaaS provider, the user is bound to the SaaS provider by agreeing with an SLA [49]. When the binding is completed, communication between the user and selected SaaS provider is done directly.

When the user sends their request to the SaaS provider,the provider, along with the cloud broker, looks for a cloud provider. Once the cloud provider is selected to handle the request, the SaaS provider binds to the selected cloud provider. Eventually, the SaaS provider presents the assumed service and makes it ready for use.

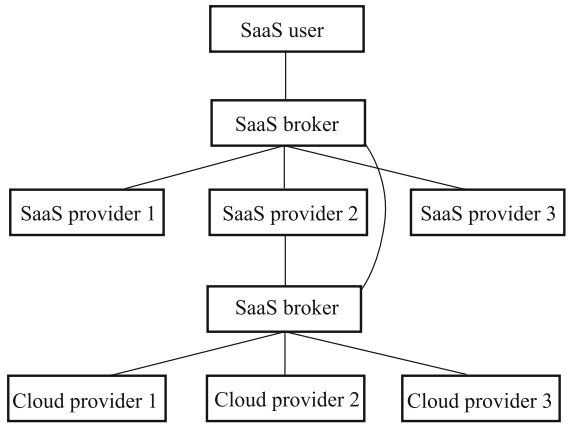


Fig. 7 Two layered brokerage[34]

Generally, the SaaS broker is in charge of binding the user to the SaaS provider, binding cloud brokers, and monitoring QoS levels in SaaS providers. The cloud broker is also responsible for binding the SaaS brokers, providing necessary information for SaaS providers, and monitoring the QoS levels.

Communication between the cloud broker and the SaaS broker is done in two phases: discovery and binding. Firstly, the necessary information is added to a registry to allow communication between brokers. One of the key pieces of information is the identity of the SaaS provider that is providing the service. Then, based on an SLA, binding is done between two brokers. Moreover, a property called efficiency is used to evaluate each SaaS provider based on supplied information from the cloud broker. Efficiency is defined as the level of service provision of the SaaS provider relative to the cloud provider's QoS level. QoS parameters that have been considered for computing efficiency are availability, reliability, and response time.

3.2.4 SMICloud

A framework is used in this approach [35] to measure quality and priority of cloud services by ranking and comprising various cloud providers based on user requirements. This framework can create a healthy competition between cloud providers to satisfy SLA and improve QoS. As mentioned before, SMI attributes can be used to provide various ser-

vices to customers. SMI attributes used in this approach are service response time, sustain ability, suitability, accuracy, transparency, inter operability, availability, reliability, stability, price, adaptability, and elasticity. It must be mentioned that there are numerous challenges in obtaining user considered quality and ranking different cloud providers, such as determining away to measure SMI attributes and ranking cloud services based on SMI attributes.

Deciding on which service matches best with all functional and non-functional requirements is an MCDM problem. In the SMICloud approach, AHP is used to solve the problem of assigning weights to features while considering interdependence between them. Ranking services take place in four phases: (1) hierarchy structure for cloud services based on SMI key performance indicators, (2) computation of relative weights of each QoS and service, (3) relative value-based weights for ranking cloud services, and (4) aggregation of relative rankings for each SMI attribute.

To take a more detailed look at this approach, were view its architectural sections:

- *SMICloud Broker*, which receives a customer's request and discovers and ranks suitable services accordingly. This section has three components: SLA management, which keeps the history of SLAs for customers and cloud providers, SMI calculator, which calculates the SMI attributes used in prioritizing cloud providers, and the ranking system, which ranks selected services by the cloud broker.
- *Monitoring* discovers cloud services that can satisfy user's essential QoS requirements. It also monitors cloud service performance.
- *Service catalogue* holds services and attributes advertised by different cloud providers.

3.2.5 Brokering for optimized placement of VMs

Designers of this approach [36] have stated two roles for brokers: they provide the scheduling mechanisms required to optimize the placement of VMs among multiple clouds and they offer a uniform management interface with operations, e.g., to deploy, monitor, and terminate VMs.

In this approach a multi-cloud architecture for cloud brokerage and VM management is suggested alongside an algorithm for optimized placement of VMs in multi-clouds. The suggested model consists of price, performance, hardware limitation, and load balance. Multi-clouds have improved performance compared to single-clouds and decrease cost. Figure 8 shows the suggested broker architecture. Considering the infrastructure criteria, limitations of the user, provider's suggestions, and scheduling algorithm used, the broker scheduler prepares an optimized deployment plan to place VMs in different clouds. The deployment plan embodies clear implementation of user requests and includes a list of VM templates. Each pattern consists of a target cloud provider to place VMs, as well as special attributes for the selected provider. The cloud scheduling problem can be solved by a static or dynamic approach. A static approach is used when requested virtual resources are limited and provider conditions such as price and availability are stable in the whole life-cycle of the cloud. In this scenario, resources are selected offline only once. A dynamic approach is used for

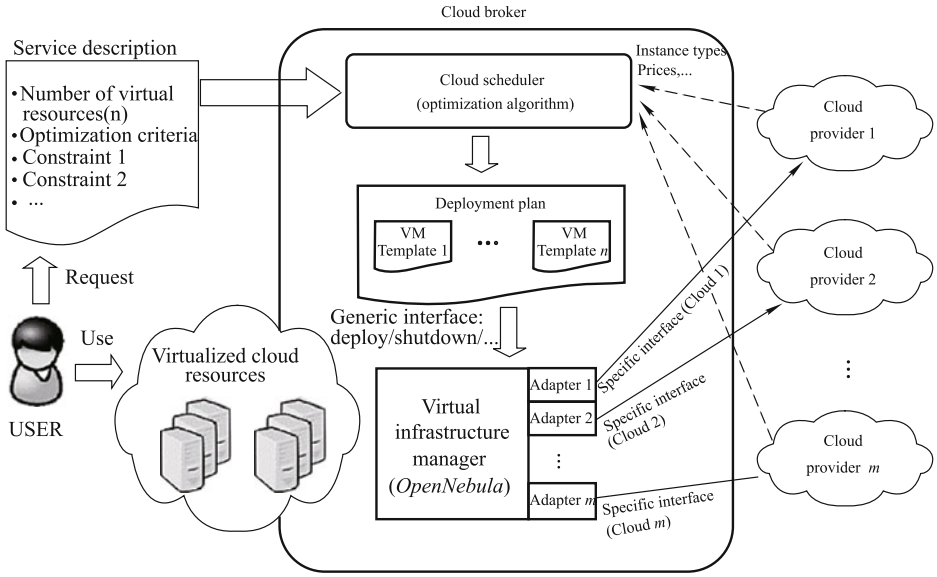


Fig. 8 Brokering for optimized placement of VMs [36]

variable-sized services, such as for a cluster server with variable required resources. Regarding the static conditions of current cloud providers, practical evaluation of this approach is also static.

VIM plays the second role of the broker by providing an abstract layer in a heterogeneous collection of clouds. This part is in charge of placing each VM in a selected cloud. VIM caters for user interaction with the virtual infrastructure by making the respective IP addresses of the infrastructure components available to the user once it has deployed all VMs. This way, the user is not informed and concerned about the distribution of resources in clouds. In this broker, VIM components are based on the OpenNebula framework [50].

3.2.6 OWL-S based semantic CSB

This approach [14] presents a service-oriented cloud broker, along with a discovery system and service selection based on OWL-S [51]. It supports dynamic semantic matching to describe cloud services with complicated limitations. It also presents an approach to resolve challenges such as support for complex constraints and dynamic discovery of services in cloud environments.

The OWL-S service description approach for displaying services has three main parts: *Service Profile* for service advertising and discovering, *Service Model* for service operation description in details, and *Service Grounding* for sending messages and allowing communication between services. SWRL [52] is also used for modeling constraints in the ontology domain.

Matching between user requests and service providers is done in five ways [53]:

- *Exact match*, where provider offers an exact service to the user request.
- *Subsume match*, where provider offers more than what the user requests.
- *Invert-subsume match*, where user has requested more than what the provider offers.
- *Partial match*, where provider can satisfy some of the user's requirements.
- *Fail match*, where provider cannot satisfy any of the user's requirements.

Figure 9 shows the architecture of this approach. Service providers offer their own services to the system. Offered services are aggregated in a repository. Suitable ranges for ontologies are defined in advance. When a service request and user preferences are received, the system starts a

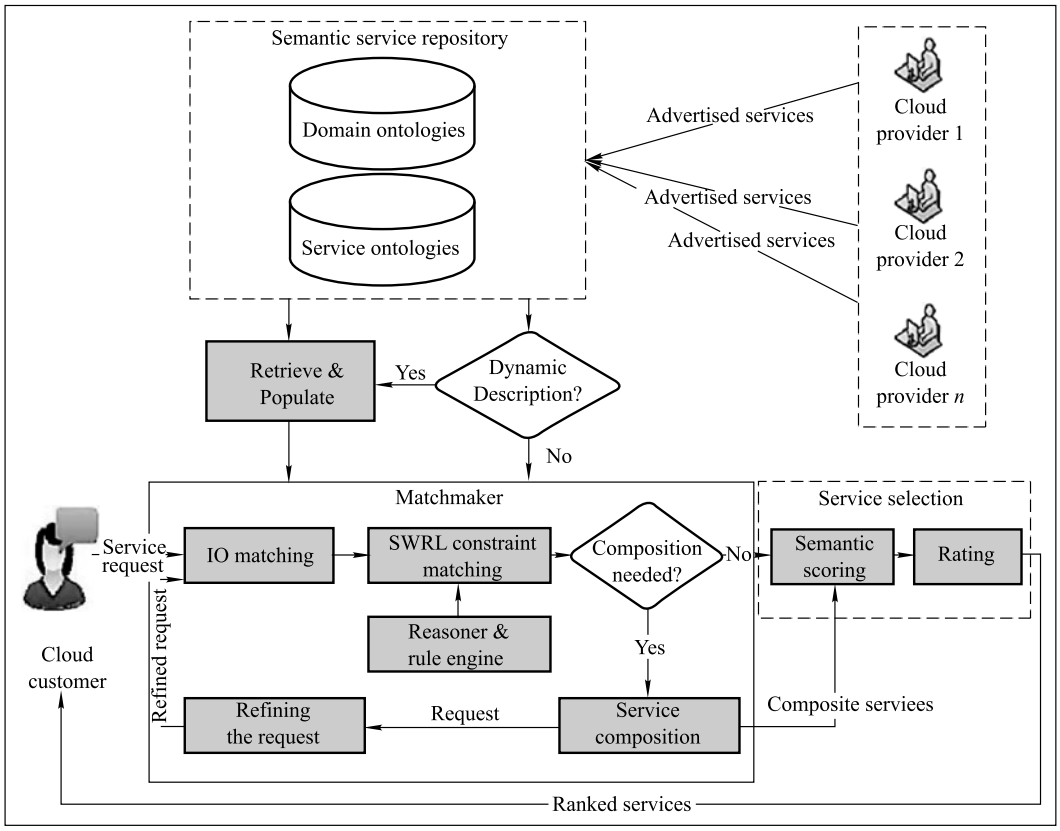


Fig. 9 OWL-S based semantic CSB [14]

matchmaking process to match offered services with services in the repository using *IOMatching* and *constraint matching*. The latter uses SWRL rule matching and a reasoning engine similar to *pellet*. This is done over all repository services and a check is performed to find whether the best obtained match is an invert-subsume. If the answer is yes, the request is refined by a service composition function and the matchmaker is re-invoked. Finally, in the *service selection* stage, discovered services are scored and ranked by a *semantic scoring* function and are returned to the user as the best-matched services.

3.2.7 Price optimization using 0-1 Knapsack

This broker [4] uses the 0-1 Knapsack [54] for pricing optimization in service integration. The designers of this broker introduced two main types of pricing model in the cloud domain: a static model, where price remains unchanged once it has been determined and a dynamic model, where price changes dynamically according to factors such as resource availability and demand. They also consider five factors for pricing: initial cost, lease period, QoS, resource age, and maintenance expenditure.

One of the key CSB functions is the service price optimization for the customer and provider. The CSB must consider all SLAs to do this for each service and to make the best choices. Some criteria such as cost, QoS, and quality of experience are considered for service integration. As a result, new SLA and services are created.

The Knapsack problem,which is used to solve pricing optimization in service integration problems, has exactly the same base structure. Applied variables are also based on fundamental cases, stating that *profit* is equal to sum of service prices, *bound* is equal to bound of the service price, and *weight* is equal to service weight.

An optimal service integration is calculated iteratively by the following three steps in order to create a state space tree:

- Visiting a node and calculating the value of each *profit* and *weight*.
- Calculating the value of each *totweight* and *bound*.
- Comparing *weight* and *W* and *bound* and *maxprofit* and determining whether the node is promising or not.

4 Main challenges

The most important challenges found in Section 3 are summarized in this section. Challenges that are common to all CSBs are reported first, followed by challenges that are specific to only some CSBs.

4.1 Common challenges

4.1.1 No support for various cloud service models

An effective CSB must be able to find suitable provider(s) for all kinds of cloud services and respond to all customer requests. Some approaches investigated in Section 3 are only capable of representing infrastructure services, i.e., IaaS. These approaches are: multi-cloud brokering [24], STRATOS [29], QBROKAGE [28], brokering using Game Theory [23], T-Broker [32], distributed cloud brokering [30], and dynamic brokering on federated clouds [31]. While SLA-based SaaS provisioning [3] is only designed for representing SaaS.

4.1.2 No desirable support for QoS attributes

An effective CSB must support all QoS attributes at the time of service selection. In other words, all effective and essential QoS attributes in service selection must be considered by a CSB in the service selection process. It is notable that a high number of QoS attributes complicates the service selection approach, which must aim to meet customers' needs conformingly. Some approaches studied in Section 3consideronly few QoS attributes. Two layered brokerage [34] only considers three attributes (availability, reliability, and response time). Multi-cloud brokering [24] only considers cost, efficiency, and energy consumption. Brokering using Game Theory [23] only considers price and quality as QoS parameters.

4.2 Specific challenges

In addition to common challenges mentioned in Section 4.1, each approach has its own specific challenges. Therefore,it is necessary for these challenges to be identified in order to help designers of future CSBs in making them effective.

In AHP & TOPSIS [21], despite a pairwise comparison matrix being used to prioritize SMI attributes, nonprofessional users are forced to define their priorities despite them being largely unfamiliar with the AHP structure. Moreover, nothing is mentioned about SLA in the AHP & TOPSIS paper.

CSP-index [33] has received lots of information from users about their requested services. But users are mostly unable to provide all types of requirements–specifically highly technical information [55]. QoS parameters used in the CSP-index

paper are not identified [18].

In SLA-based SaaS provisioning [3], when a provider selected by these lection manager cannot satisfy the user requirements, the SLA manager related to that provider sends another suggestion to the CSB SLA manager. However, since user preferences are not prioritized, new suggestions may not satisfy user preferences, which can result in a highly time consuming service selection process. Moreover, QoS attributes considered in this method are not clearly determined.

In Multi-cloud brokering [24], the service description file to be prepared by the user has various sections that can be filled out only by experienced and expert users.

Because user feedback is a main aspect of T-Broker [32], attracting users to trust the system and encouraging them to send their feedback are challenging. Also, the proposed approach does not discuss ways of controlling malicious behaviors [56].

Distributed cloud brokering [30] prioritizes constraints according to user preferences and classifies preferences as hard or soft. However,neither hard preferences nor soft preferences are prioritized in their own category.

Two layered brokerage [34] does brokering in two layers, namely IaaS and SaaS layers, resulting in overlay communications on the network. In addition, SaaS and IaaS providers share the same functionality under a broker, and each broker can only handle requests with a specific single functionality.

SMICloud [35] uses AHP to compare and rank service providers. It does not automate the selection process and requires significant user input, which hinders its usage in practice [57]. This approach can beused only for quantifiable QoS attributes and it is not suitable for non-quantifiable QoS attributes [17].

STARTOS [29] does not consider QoS attributes at all. The geographical locations of the serving data centers are not considered either, which makes it impossible to allow legislation-aware application brokering [58].

QBROKAGE [28] exploits only the limited information that commercial providers are likely to make available to customers, such as VM costs and characteristics in terms of storage, memory, etc. Moreover, its current version does not support elasticity for an already deployed application.

Brokering for optimized placement of VMs [36] only considers static scenarios where users' and providers' conditions do not change over a long time period and the placement

Table 2 CSB approaches (single service model)

Approaches	Main contributions	Main challenges
QBROKAGE [28]	1. Modeling applications, resources, and QoS attributes in OVF format 2. Making an application graph based on OVF file 3. Using game theory to map applications to providers	1. Only uses commercial provider information 2. Is not elastic to support previously deployed applications
SLA-based SaaS provisioning [3]	1. Using QoS attributes for ranking SaaS providers 2. Using multi-attributes negotiation model based on SLA agreement between customer and provider	1. QoS attributes are not determined clearly 2. Lengthy service selection time due to the lack of user prioritization
Brokering using Game Theory [23]	1. Receiving and analyzing user requests using history 2. Using game theory in SLA negotiations to gain useful agreements between providers and customers	1. Only cost and quality parameters are considered in selection service 2. Lengthy negotiation time when the number of SLA parameters increases
STRATOS [29]	1. Using RAD as a multi-criterion optimization problem 2. Receiving TDF file from the user to determine the topology and application goals 3. Identifying possible configuration and goals of optimization	1. No use of QoS attributes 2. No ability to implement legislation aware brokering because of ignoring geographic datacenter placement
Distributed cloud brokering [30]	1. Using CFP to appoint SLA template and exchanging policies 2. Using distributed brokering	User preferences are placed into hard and soft classes. If user selects a few parameters in one class, the broker doesn't distinguish between them
Multi-cloud brokering [24]	1. Using a scheduler to deploy VMs in clouds 2. Using a VM manager for monitoring and updating the VM list 3. Using a cloud manager for updating the instance list and dynamic pricing	1. Only price, efficiency, and energy consumption are considered in service selection 2. Receiving much professional information from users in the service description file
Dynamic brokering on federated clouds [31]	1. Using multi-agent systems in federated clouds 2. Negotiating in three ways: consumer agent, RBA communication, and RPA communication	1. Suitable only for federated clouds 2. Only price is considered for service selection
T-Broker [32]	1. Using a trust-aware architecture in multi-cloud environment 2. Using Sensor-Based service monitoring for monitoring the real-time service data to guarantee the SLA 3. Using virtual infrastructure manager 4. Using a lightweight feedback mechanism	1. Encourages users to send their feedback to ensure their trust 2. Control of malicious behavior is not determined

Table 3 CSB approaches (Multiple service model)

Approaches	Main contributions	Main challenges
AHP & TOPSIS [21]	1. Identifying suitable criteria 2. Evaluating the weight of criteria using AHP 3. Ranking cloud providers using TOPSIS	1. It is not clear how users declaretheir priorities to the system 2. A special structure is not defined for SLA
CSP-index [33]	1. Indexing Cloud Service Providers by: a) Property and relationship encoding b) Index key generation 2. Query Algorithm for Cloud Service Selection with: a) Query definition and encoding b) K-nearest neighbor search c) Refinement and consideration of special criteria	1. Receiving much expert information from the user 2. QoS attributes are not determined clearly
Two layers brokering [34]	1. Using two brokers: SaaS broker and cloud broker 2. To create communication between brokers in two phases: discovery and binding 3. To define the*efficiency* property based on cloud broker information to assess SaaS providers	1. Only availability, reliability, and response time are considered in service selection 2. High rate of network communications when the number of SaaS and IaaS providers is high
SMICloud [35]	1. Discovery and ranking services by AHP by SMICloud Broker component 2. Monitoring the performance of allocated services by a monitoring component 3. Service maintenance and itscharacteristics using a service catalogue component	1. Receiving muchinformation from the user non-automatically 2. Lower efficiency compared to non-quantifiable QoS
Brokering for optimized placement of VMs [36]	1. Using cloud scheduler to supply an optimized deployment plan for VMs 2. Using a virtual infrastructure manager to deploy each VM on a selected cloud	1. It is not quantifiable in case of dynamic service allocation 2. Differences in the performances of virtual machine managers is not considered
OWL-S based semantic CSB [14]	1. Using dynamic semantic matching, discovery, and service selection based on OWL-S 2. Using semantic scoring function for scoring and ranking discovered services	1. QoS attributes in service selection are not considered. 2. It cannot work with specific ontologies of some providers
Price optimization using 0-1 Knapsack [4]	Using 0-1 Knapsack for pricing optimization in cloud service integration	1. The only criteria to integrate services is the price 2. Uses time-consuming algorithmsto deal with various providers with different SLA parameters

decision is made only once [59]. It also presents fine-grained inter operability of cloud services by way of an API that does not take into account possible differences in virtual machine managers that are deployed in the infrastructures of different cloud providers [60].

Dynamic brokering on federated clouds [31] is only suitable for federated clouds, and thus, is not useful in multi-cloud domains. It appears to use price as the main criteria in its service selection and is not clear as to which other QoS parameters can be considered.

OWL-S based semantic CSB [14] does not consider the service QoS metrics and characteristics in its selection process [61, 62]. This may pose challenges if part of the service description is in free text or if some cloud providers adopt customized ontologies [26].

Price optimization using 0-1 Knapsack [4] considers price as the only criterion to offer a service from composition of a number of services from possibly different providers. Sole reliance on price is not a proper criterion for service selection. Furthermore, recursively searching for providers and their services with different SLAs and finding suitable services could be very time consuming.

Brokering using Game Theory [23] uses a quality parameter that is too general to be considered as an effective and useful parameter in SLA. Moreover, the higher the number of SLA providers, the longer negotiating takes using the game theory; it takes longer for the game to reach equilibrium over more rounds.

5 Evaluation and comparison

This section compares and evaluates cloud service selection approaches by CSBs noted in Section 3. Tables 2 and 3 summarize the main principals of each approach and their main challenges. Since all stated approaches in Table 2 support only one cloud service model, the type of supported service in the main challenges column is not stated. Table 4 compares all represented approaches based on the characteristics of an effective CSB that we proposed in Section 2.

Table 4 Comparison of represented approaches based on the proposed attributes of an effective CSB

Approaches	Qos attributes used	SLA violation determination	Price optimizing mechanism	Cloud service model selection	Using user profile	Dynamic/ Static brokering	Quantity & quality of information received from user	Prioritize customer preferences
QBROKAGE	Commercial providers QoS	No	No	IaaS	No	D	Balanced	No
SLA-based SaaS provisioning	Not determined clearly	Yes	No	SaaS	Yes- by profile repository	D	Balanced	No
Brokering using Game Theory	Cost & Quality	No	Yes- by Payoff function	IaaS	Yes- by History component	D	Balanced	No
STRATOS	None	No	Yes	IaaS	No	S	Balanced	Yes
Distributed cloud brokering	Various attributes	No	No	IaaS	Yes	D	Balanced	Incomplete- only hard and soft groups
Multi-cloud brokering	Cost, Performance & Energy consumption	No	Yes	IaaS	Yes- by historical profile of the deployed service	D	Too Muchand Professional	No
Dynamic brokering on federated clouds	Cost	No	No	IaaS	No	D	Balanced	No
T-Broker	Various attributes	No	No	IaaS	No	D	Balanced	No
AHP & TOPSIS	SMI attributes	No	No	Multiple	No	S	Too Much and Professional	Yes- by AHP
CSP-index	Not determined clearly	No	No	Multiple	No	S	Too Much and Professional	No
Two layers brokering	Availability, Reliability & Response time	No	No	Multiple	No	D	Balanced	No
SMICloud	SMI attributes	No	No	Multiple	Yes- by SLA Manager and Monitoring components	S	Too Much and Professional	Yes- by AHP
Brokering for optimized placement of VMs	Performance, Cost & Load balance	No	Yes	Multiple	No	S	Balanced	No
OWL-S based semantic CSB	None	No	No	Multiple	No	D	Balanced	No
Price optimization using 0-1 Knapsack	Not determined clearly	No	Yes- using 0-1 Knapsack	Multiple	No	S	Balanced	No

6 Conclusion

This paper reviewed the issue of cloud service selection by considering different CSB approaches. Eight attributes for an effective CSB approach were proposed: (1) using various QoS attributes, (2) SLA violation recognition mechanism, (3) price optimization mechanism, (4) selecting suitable provider for different types of cloud services, (5) using customer profile to allocate later services, (6) dynamic brokering, (7) not receiving all service information from the customer, and (8) prioritizing customer preferences. Most of the prominent works from recent years were reviewed, main challenges were stated for each one, and they were comparatively tabulated with respect to the proposed attributes. No investigated CSB approach ideally supported all the proposed

attributes for an effective CSB completely. Though it may be possible in theory, it is potentially impractical and unnecessary to have a full-fledged CSB supporting all attributes. Therefore, CSB designers must prioritize the proposed attributes for an effective CSB in order to find a logical tradeoff between different attributes and select the ones that conform best with the main objectives of the CSB to be designed.

References

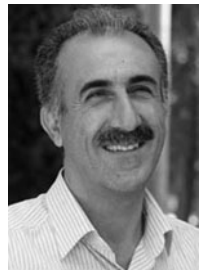
1. Mell P, Grance P. The NIST Definition of Cloud Computing. NIST Special Publication 800-146, 2011
2. Wadhwa B, Jaitly A, Hasija N, Suri B. Cloud service brokers: addressing the new cloud phenomenon. In: Rajsingh E B, Bhojan A, Peter J D, eds. Informatics and Communication Technologies for Societal Development, Springer International Publishing, 2015, 29–40
3. Badidi E. A cloud service broker for SLA-based SaaS provisioning. In: Proceedings of International Conference on Information Society. 2013, 61–66
4. Shin Y R, Huh E N. Optimization for reasonable service price in broker based cloud service environment. In: Proceedings of the 4th International Conference on Innovative Computing Technology. 2014, 115–119
5. Buyya R, Ranjan R, Calheiros R N. InterCloud: utility-oriented federation of Cloud computing environments for scaling of application services. In: Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing. 2010, 13–31
6. Ferrer A J, Hernández F, Tordsson J, Elmroth E, Ali-Eldin A, Zsigri C, Sirvent R, Guitart J, Badia R M, Djemame K, Ziegler W. OPTIMIS: a holistic approach to cloud service provisioning. Future Generation Computer Systems, 2012, 28(1), 66–77
7. Simarro J L L, Moreno-Vozmediano R, Montero R S, Llorente I M. Dynamic placement of virtual machines for cost optimization in multi-cloud environments. In: Proceedings of International Conference on High Performance Computing and Simulation. 2011, 1–7
8. Liu F, Tong J, Mao J, Bohn R, Messina J, Badger L, Leaf D. NIST Cloud Computing Reference Architecture. NIST Special Publication 500, 2011
9. Grozev N, Buyya R. Inter-cloud architectures and application brokering: taxonomy and survey. Software: Practice and Experience, 2014, 44(3): 369–390
10. Sanchez F D, Al Zahr S, Gagnaire M, Laisne J P, Marshall I J. CompatibleOne: bringing cloud as a commodity. In: Proceedings of IEEE International Conference on Cloud Engineering. 2014, 397–402
11. Bhattacharya A, Choudhury S. Service insurance: a new approach in cloud brokerage. In: Chaki R, Saeed K, Choudhury S, et al. eds. Applied Computation and Security Systems. Springer International Publishing, 2015, 39–52
12. Song H, Bae C S, Lee J W, Youn C H. Utility adaptive service brokering mechanism for personal cloud service. In: Proceedings of Military Communications Conference. 2011, 1622–1627
13. Somasundaram T S, Govindarajan K, Rajagopalan M, Rao S M. A broker based architecture for adaptive load balancing and elastic resource provisioning and deprovisioning in multi-tenant based cloud environments. In: Proceedings of International Conference on Advances in Computing. 2012, 561–573
14. Ngan L D, Kanagasabai R. Owl-s based semantic cloud service broker. In: Proceedings of the 19th IEEE International Conference on Web Services. 2012, 560–567
15. Whaiduzzaman M, Haque M N, Rejaul Karim Chowdhury M, Gani A. A study on strategic provisioning of cloud computing services. The Scientific World Journal, 2014, 1–16
16. Sun L, Dong H, Hussain F K, Hussain O K, Chang E. Cloud service selection: state-of-the-art and future research directions. Journal of Network and Computer Applications, 2014, 45: 134–150
17. Subha M, Banu M U. A survey on QoS ranking in cloud computing. International Journal of Emerging Technology and Advanced Engineering, 2014, 4(2): 482–488
18. Jula A, Sundararajan E, Othman Z. Cloud computing service composition: a systematic literature review. Expert Systems with Applications, 2014, 41(8): 3809–3824
19. Chao K M, Anane R, Chen J H, Gatward R. Negotiating agents in a market-oriented grid. In: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid. 2002, 436
20. Kertész A, Kecskemeti G, Brandic I. An interoperable and self-adaptive approach for SLA-based service virtualization in heterogeneous Cloud environments. Future Generation Computer Systems, 2014, 32: 54–68
21. Achar R, Thilagam P S. A broker based approach for cloud provider selection. In: Proceedings of International Conference on Advances in Computing, Communications and Informatics. 2014, 1252–1257
22. Schaffer H E. X as a service, cloud computing, and the need for good judgment. IT Professional, 2009, 11(5): 4–5
23. Ray B K, Khatua S, Roy S. Negotiation based service brokering using game theory. In: Proceedings of Applications and Innovations in Mobile Computing Conference. 2014, 1–8
24. Simarro J L L, Aniceto I S, Moreno-Vozmediano R, Montero R S, Llorente I M. A cloud broker architecture for multi-cloud environments. In: Proceedings of Large Scale Network-Centric Distributed Systems Conference. 2013, 359–376
25. Brock M, Goscinski A. Enhancing Cloud Computing Environments using a Cluster as a Service. New York: Wiley Press, 2011
26. Parhi M, Pattanayak B K, Patra M R. A multi-agent-based framework for cloud service description and discovery using ontology. In: Jain L C, Patnaik S, Ichalkaranje N, eds. Intelligent Computing, Communication and Devices, Springer International Publishing, 2015, 337–348
27. Lee Y T, Wu C S. A quality-based semantic service broker using reachability indexes. In: Proceedings of IEEE World Forum on Internet of Things. 2014, 277–282
28. Anastasi G F, Carlini E, Coppola M, Dazzi P. QBROKAGE: a genetic approach for QoS cloud brokering. In: Proceedings of the 7th IEEE International Conference on Cloud Computing. 2014, 304–311
29. Pawluk P, Simmons B, Smit M, Litoiu M, Mankovski S. Introducing STRATOS: a cloud broker service. In: Proceedings of the 5th IEEE International Conference Cloud Computing. 2012, 891–898
30. Amato A, Di Martino B, Venticinque S. A distributed cloud brokering

service. Informatica, 2015, 26(1): 1–15

31. Haresh M, Kalady S, Govindan V. Agent based dynamic resource allocation on federated clouds. In: Proceedings of IEEE Conference on Recent Advances in Intelligent Computational Systems. 2011, 111–114
32. Li X, Ma H, Zhou F, Yao W. T-Broker: a trust-aware service brokering scheme for multiple cloud collaborative services. IEEE Transactions on Information Forensics and Security, 2015, 10(7): 1402–1415
33. Sundareswaran S, Squicciarini A, Lin D. A brokerage-based approach for cloud service selection. In: Proceedings of the 5th IEEE International Conference on Cloud Computing. 2012, 558–565
34. Lim E, Thiran P. Communication of technical QoS among cloud brokers. In: Proceedings of IEEE International Conference on Cloud Engineering. 2014, 403–409
35. Garg S K, Versteeg S, Buyya R. SMICloud: a framework for comparing and ranking cloud services. In: Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing. 2011, 210–218
36. Tordsson J, Montero R S, Moreno-Vozmediano R, Llorente I M. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Generation Computer Systems, 2012, 28(2): 358–367
37. Anastasi G F, Carlini E, Coppola M, Dazzi P. Smart cloud federation simulations with CloudSim. In: Proceedings of the 1st ACM Workshop on Optimization Techniques for Resource Management in Clouds. 2013, 9–16
38. Ye Z, Zhou X, Bouguettaya A. Genetic algorithm based QoS-aware service compositions in cloud computing. In: Proceedings of the 16th International Conference on Database systems for advanced applications. 2011, 321–334
39. Li X, Yang Y. Trusted data acquisition mechanism for cloud resource scheduling based on distributed agents. China Communication, 2011, 8(6): 108–116
40. Smit M, Pawluk P, Simmons B, Litoiu M. A web service for cloud metadata. In: Proceedings of the 8th IEEE World Congress Services. 2012, 361–368
41. Shoham Y, Leyton-Brown K. Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge: Cambridge University Press, 2008
42. Li X, Zhou F. PG-TRUST: a self-adaptive and scalable trust computing model for large-scale peer-to-peer grid computing. International Journal of Software Engineering and Knowledge Engineering, 2011, 21(5): 667–692
43. Yoon K P, Hwang C L. Multiple attribute decision making: an introduction. Sage Publications, 1995
44. Siegel J, Perdue J. Cloud services measures for global use: the service measurement index (SMI). In: Proceedings of SRII Global Conference. 2012, 411–415
45. Saaty T L. How to make a decision: the analytic hierarchy process. European Journal of Operational Research, 1990, 48(1): 9–26
46. Kalepu S, Krishnaswamy S, Loke S W. Verity: a QoS metric for selecting Web services and providers. In: Proceedings of the 4th Conference on Web Information Systems Engineering Workshops. 2003, 131–139
47. AlZain M, Pardede E, Soh B, Thom J. Cloud computing security: from single to multi-clouds. In: Proceedings of the 45th International Conference on System Science. 2012, 5490–5499
48. Vukolić M. The Byzantine empire in the intercloud. ACM SIGACT News, 2010, 41(3): 105–111
49. Alhamad M, Dillon T, Chang E. Conceptual SLA framework for cloud computing. In: Proceedings of the 4th IEEE International Conference on Digital Ecosystems and Technologies. 2010, 606–610
50. Sotomayor B, Montero R S, Llorente I M, Foster I. Virtual infrastructure management in private and hybrid clouds. IEEE Internet Computing, 2009, 13(5): 14–22
51. Aslam M A, Auer S, Shen J, Herrmann M. Expressing business process models as OWL-S ontologies. In: Proceedings of Business Process Management Workshops. 2006, 400–415
52. Gonzalez-Castillo J, Trastour D, Bartolini C. Description logics for matchmaking of services. In: Proceedings of Workshop on Applications of Description Logics. 2002
53. Ngan L D, Tsai Flora S, Keong C C, Kanagasabai R. Towards a common benchmark framework for cloud brokers. In: Proceedings of the 18th IEEE International Conference on Parallel and Distributed Systems. 2012, 750–754
54. Neapolitan R, Naimipour K. Foundations of Algorithms. Swdbury, Mass: Jones & Bartlett Publishers, 2010
55. Karim R, Ding C, Miri A. An end-to-end QoS mapping approach for cloud service selection. In: Proceedings of the 9th IEEE World Congress on Services. 2013, 341–348
56. Komninos N, Junejo A K. Privacy preserving attribute based encryption for multiple cloud collaborative environment. In: Proceedings of the 8th IEEE/ACM International Conference on Utility and Cloud Computing. 2015, 595–600
57. Juan-Verdejo A, Zschaler S, Surajbali B, Baars H, Kemper H G. InCLOUDer: a formalized decision support modelling approach to migrate applications to cloud environments. In: Proceedings of the 40th EUROMICRO Conference on Software Engineering and Advanced Applications. 2014, 467–474
58. Grozev N, Buyya R. Multi-cloud provisioning and load distribution for three-tier applications. ACM Transactions on Autonomous and Adaptive Systems, 2014, 9(3): 13
59. Simarro J L L, Moreno-Vozmediano R, Montero R S, Llorente I M. Scheduling strategies for optimal service deployment across multiple clouds. Future Generation Computer Systems, 2013, 29(6): 1431–1441
60. Amato A, Di Martino B, Venticinque S. Evaluation and brokering of service level agreements for negotiation of cloud infrastructures. In: Proceedings of International Conference Internet Technology and Secured Transactions. 2012, 144–149
61. Afify Y M, Moawad I F, Badr N L, Tolba M. Cloud services discovery and selection: survey and new semantic-based system. In: Hassanien A E, Kim T H, Kacprzyk J, et al. eds. Bio-inspiring Cyber Security and Cloud Services: Trends and Innovations, Springer International Publishing, 2014. 449–477
62. Afify Y M, Moawad I F, Badr N L, Tolba M. A semantic-based software-as-a-service (SaaS) discovery and selection system. In: Proceedings of the 8th International Conference on Computer Engineering & Systems. 2013, 57–63

Meysam Vakili received his BS degree in software engineering from University of Fasa, Fars, Iran in 2011, and MS degree in software engineering from University of Science and Culture, Tehran, Iran in 2014. His research interests include distributed systems, Internet of Things, cloud computing, and complex event processing.

Neda Jahangiri received her BS degree in software engineering in 2012, and MS degree in software engineering from University of Science and Culture, Tehran, Iran in 2015. Her research interests include cloud computing and databases.



Mohsen Sharifi is a professor of System Software Engineering at the School of Computer Engineering at Iran University of Science and Technology, Iran. He directs a distributed systems research group and laboratory. His main interest is in the development of distributed systems, solutions, and applications, particularly for use in various fields of science. The development of a true distributed operating system is on top of his wish list. He received his BS, MS, and PhD degrees in computer science from the University of Manchester, UK.