Supporting Users in Cloud Plan Selection

Sabrina De Capitani di Vimercati, Sara Foresti, Giovanni Livraga, Vincenzo Piuri, and Pierangela Samarati

Università degli Studi di Milano, 26013 Crema, Italy
firstname.lastname@unimi.it

Abstract. Cloud computing is a key technology for outsourcing data and applications to external providers. The current cloud market offers a multitude of solutions (plans) differing from one another in terms of their characteristics. In this context, the selection of the right plan for outsourcing is of paramount importance for users wishing to move their data/applications to the cloud. The scientific community has then developed different models and tools for capturing users' requirements and evaluating candidate plans to determine the extent to which each of them satisfies such requirements. In this chapter, we illustrate some of the existing solutions proposed for cloud plan selection and for supporting users in the specification of their (crisp and/or fuzzy) needs.

Keywords: Cloud computing, cloud plan selection, user requirements, fuzzy logic

1 Introduction

The cloud providers offer today a large, rich, and diversified set of services on which users can rely to store their data and deploy their applications. Usually, such services are proposed in terms of pre-defined configurations (plans) with different features that make, for example, a solution more suitable for data storage, another for the deployment of performant applications, and so on. This can be easily observed by a simple look at the current panorama, where cloud providers (e.g., Amazon) offer a plethora of different plans (e.g., S3, EC2, just to mention a few). Although the richness and diversity of the current cloud market can be beneficial to users since, the more the possible options, the more each user will be able to find a plan well-aligned to her needs, the selection of a plan among those available in the market can be a difficult task that requires to address several problems. First, there is the need to determine the parameters that can be used to evaluate and compare candidate plans and to select the right one. Typically, every provider publishes Service Level Agreements (SLAs), which are binding contracts that specify minimum guarantees on Quality of Service (QoS) parameters ensured during service provision. For instance, SLAs include the minimum uptime percentage that is guaranteed, together with indications on the possible compensations that the user can get if such minimum level is not met. However, since there is not a general template for SLA definition, different SLAs can

include different information, or even the same information but with different names (e.g., 'monthly uptime' in Amazon's Compute SLA and 'monthly availability' in Rackspace's Cloud SLA). Hence, while it can seem natural to look at parameters declared in SLAs to compare cloud plans for their assessment and selection, the task can be very complex. A second problem consists in identifying a way to actually perform the assessment of cloud plans. In this case, the optimization criteria to be met can be multiple and possibly contrasting: as an example, the cheapest plan might not be the most performant, and yet a user might want to select a plan which maximizes performance while minimizing cost. Orthogonally to these problems, another issue relates to providing support to users in the specification of their requirements to be taken into account in the assessment and selection of cloud plans. Different users might have different (and possibly contrasting) needs to be considered, due to, for example, laws, regulations, or simply due to the specific applicative scenario. Having means and techniques for allowing users to specify arbitrary requirements and for enforcing them is therefore fundamental for responding to users' desiderata.

The scientific community has devoted many efforts to study and design solutions for the general problem of secure data management (e.g., [28, 29]), also focusing on the cloud plan selection problem thus generating solutions to: *i)* define standardized sets of attributes and/or metrics over which evaluate a candidate plan (e.g., [4, 18]); *ii)* evaluate multiple/conflicting requirements (e.g., [8, 9])s; and *iii)* support users in a friendly and easy specification of their needs (e.g. [6, 12, 17]). In this chapter, we present some of the existing models and solutions proposed for addressing all these aspects.

The remainder of this chapter is organized as follows. Section 2 illustrates existing techniques for identifying attributes to be used for selecting and assessing cloud plans. Section 3 focuses on the problem of supporting users towards a flexible and user-friendly specification of requirements and preferences that should be taken into account in cloud plan selection. Section 4 overviews the possible use of fuzzy logic in cloud plan selection for specifying user requirements. Finally, Section 5 concludes the chapter.

2 Attributes identification

The problem of cloud plan selection requires to analyze the characteristics of the plans available in the market to determine the ones that can be considered acceptable (or more appealing) than others for outsourcing. For instance, the selection of a plan for outsourcing mission-critical but non-sensitive data might consider optimal a plan that ensures maximum availability. In this section, we first illustrate some of the existing solutions that rely on Quality of Service (QoS) evaluation (Section 2.1), and then discuss proposals that focus on specific aspects of the problem such as QoS values predictions, dependencies management, and security parameters (Sections 2.2-2.4).

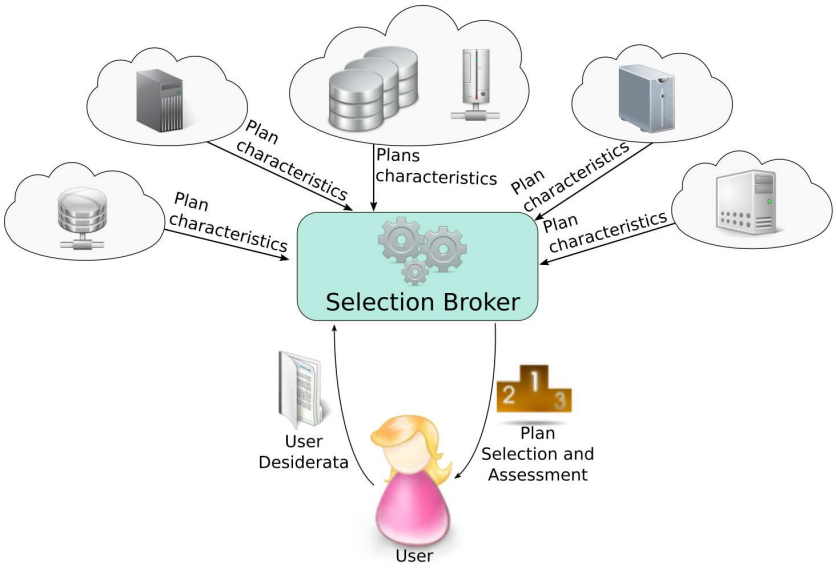

Fig. 1. Brokerage-based cloud plan selection

2.1 Quality of Service (QoS) evaluation

The most simple approach for assessing, and hence selecting, cloud plans requires to evaluate its low-level characteristics (e.g., CPU and network throughput). Typically, the most relevant characteristics considered in the analysis of cloud plans include cost, which should be low, and performance, which should be high. CloudCmp [18] compares the performance and cost of different cloud providers. CloudCmp first identifies common services offered by different cloud providers (i.e., elastic computing, persistent storage, and networking services) and then identifies the performance and cost metrics according to which such common services are compared. The values for these metrics are computed with a combination of benchmarking tasks (for elastic computing and persistent storage) and service invocations through standard tools such as ping (for networking services).

Besides the natural need for a performant plan (possibly at affordable cost), users might have more complex requirements, identifying, for example, minimum levels for different QoS attributes ensured by a provider during service provision. The solutions proposed in this context are typically based on the presence of a middleware in the system architecture playing the role of a *broker* [14], which can be trusted or verified for behavior correctness [19]. Figure 1 illustrates a typical broker-based cloud plan selection process: the selection broker is in charge of collecting both user's desiderata and plans' characteristics (possibly expressed in a machine-readable format [27]), reasoning over them, and returning to the user the result of its assessment.

Attribute	Example of sub-attributes
Accountability	Auditability, Compliance to standards, Environmental sustainability
Agility	Elasticity, Portability, Flexibility
Assurance	Reliability, Resiliency
Cost	Acquisition cost, On-going cost
Performance	Throughput, Efficiency
Security and Privacy	Measures for confidentiality, integrity, availability
Usability	Ease of usage, Ease of installation

Fig. 2. SMI attributes and an example of their sub-attributes

There have been recent efforts, by both the academia and international standardization bodies, towards the definition of a standardized set of QoS attributes that could be used by users to formulate requirements. For instance, the Cloud Service Measurement Index Consortium (CSMIC) has identified a set of QoS attributes and sub-attributes, organized in a hierarchical way, composing the *Service Measurement Index* (SMI) [14]. Figure 2 lists the seven higher-level SMI attributes and, for each of them, possible sub-attributes that contribute to it. For instance, high-level attribute *cost* depends on two sub-attributes *acquisition cost* and *on-going cost*, meaning that the cost associated with a certain cloud plan is influenced by both the cost to acquire cloud resources, and the cost to maintain and use them (e.g., communication, storage, and computation costs charged by the provider). The SMI attributes form the basis over which the proposal in [14] compares and ranks cloud plans. User requirements set bounds to the values that the attributes of interest to the user can assume, and the values assumed by plans (harvested by a broker) are evaluated against such requirements. Such an evaluation is however complex as it can also require to solve conflicts: for instance, when assessing two plans P_1 and P_2, it might happen that P_1 is better than P_2 for an attribute (say, cost) and worse than P_2 for another attribute (say, performance). To solve these issues, in [14] the authors propose to adopt a Multi-Criteria Decision Method (MCDM) that, among alternative solutions, identifies the one that optimizes a set of objective functions [2, 7, 26] (e.g., minimize cost while maximizing performance).

The proposal in [16] adopts a hybrid MCDM-based approach to select cloud plans, which combines two well-known techniques (AHP-Analytic Hierarchy Process, and TOPSIS-Technique for Order of Preference by Similarity to Ideal Solution) to reason over QoS attributes and values. MCDM, possibly coupled with machine learning, has also been proposed to select the *instance type* (i.e., the configuration of computing, memory, and storage capabilities) enjoying the best trade-off between economic costs and performance while satisfying user requirements (e.g., [23, 30]). For each of the resources to be employed (e.g., memory and CPU), these proposals select the provider (or set thereof) to be used for its provisioning as well as the amount of the resource to be obtained from each of them, so to satisfy user requirements.

QoS evaluation has also been adopted in combination with other criteria for cloud plan selection (e.g., subjective assessments and personal experience [10,

15, 24, 33]) as well as with other reasoning techniques (e.g., fuzzy logic [5, 11, 22], as we will illustrate in Section 4), and consensus-based voting techniques (e.g., [2]).

2.2 QoS prediction

The values assumed by a cloud plan for QoS attributes are usually harvested by brokers from the SLAs published by cloud providers. However, it should not be forgotten that the interaction between a user and a cloud platform operates through an Internet connection. For this reason, the values declared by the provider (*provider-side QoS*) can differ from those observed by a user (*user-side QoS*). Also, different users can observe different user-side QoS values for the same plan. For instance, the response time experienced by two different users might be different if they are located in different geographical areas or if they have access to networks with different latencies. Therefore, assessing cloud plans only based on provider-side QoS might fall short in real-world scenarios, as the criteria over which the selection operates might not consider what is actually locally observed by the user. To overcome this problem, some techniques introduced the idea of selecting cloud plans based on the user-side values of QoS attributes (e.g., [34]). A precise evaluation of user-side QoS values can however be a difficult task, as it can require actual invocations and/or usage of cloud services, causing both communication overhead and economic charges. Moreover, due to the possible differences in the values observed by different users, the same plan might be assessed differently by different users. A possible solution to this issue can consider past usage experiences of 'similar users' (e.g., users expecting to observe similar values). Measured or estimated QoS parameters are finally used to rank all the (functionally equivalent) providers among which the user can choose (e.g., [34]).

2.3 Dependencies management

Recent lines of work have investigated the problem of supporting users in specifying *arbitrary requirements* that can be considered in cloud plan selection and in SLA definition (e.g., see Section 3). Recent approaches have specifically proposed the definition of a brokering service in charge of interpreting requirements on arbitrary attributes, and of querying candidate providers on their satisfaction [9, 32]. However, when using arbitrary attributes, it may happen that certain service guarantees can be satisfied by a provider only if other conditions (maybe even insisting at the user side) are also satisfied. This is because there might be some *dependencies* among conditions: for example, the response time of a system may depend on the incoming request rate (i.e., the number of incoming requests per second). In a scenario where the user is free to set arbitrary conditions on the response time of a service, the process of evaluating requirements should carefully consider whether a candidate provider is able to respect such a requirement only if an upper bound is enforced on the number of requests per time unit. Note that, clearly, different providers/plans might entail different dependencies (e.g., two plans with different hardware/software configurations might

accept different request rates to guarantee the same response time). This clearly further complicates the cloud plan selection problem. Recent approaches have designed solutions for negotiating an SLA between a user and a cloud provider based on generic user requirements and on the automatic evaluation of dependencies existing for the provider (e.g., [9]). The solution in [9] takes as input a set of generic user requirements and a set of dependencies for a provider, and determines (if any) a *valid* SLA (vSLA) that satisfies the conditions expressed by the user as well as further conditions possibly triggered by dependencies. With reference to the example above, if the user requirements include a condition over the response time, the generated vSLA will also include a condition on the maximum supported request rate. Given a set of requirements and a set of dependencies, different valid SLAs might exist. The approach in [8] extends the work in [9] by allowing users to specify preferences over conditions that can be considered for selecting, among the valid SLAs, the one that the user prefers. Preferences are expressed over the values that can be assumed by the attributes involved in requirements and dependencies (e.g., response time and request rate). Building on the approach proposed in [9], these preferences are used to automatically evaluate vSLAs, ranking higher those that better satisfy the preferences of the user.

2.4 Security parameters

Security is undoubtedly a key requirement for many users when moving to the cloud since, by delegating the management of their resources to an external provider, they lose control over them. The selection of the cloud provider offering the best plan with respect to the required needs should then be based also considering the security guarantees ensured during service provision.

In the context of cloud service provision, security is typically guaranteed by providers through the adoption of certifications that are based on established standards, possibly specifically designed for the cloud environment [20]. Among cloud-specific solutions, the Cloud Security Alliance Cloud Controls Matrix (CSA CCM) [4] is a framework designed to provide security concepts and principles to cloud providers and to allow users to assess the security risks associated with a provider. The CSA CCM organizes concepts and principles in domains including, for example, application & interface security, identity & access management, and encryption & key management. For each domain, the CCM introduces a set of security principles: for example, a principle within domain 'encryption & key management' is 'keys must have identifiable owners (binding keys to identities) and there shall be key management policies'. With each principle, the CCM identifies the security standards and regulations whose satisfaction requires the implementation of the principle. By verifying the satisfaction of the principles declared by a provider, a user can evaluate the security guarantees of the plans offered by the provider. The Cloud Controls Matrix is well aligned to the Cloud Security Alliance guidance as well as to the Consensus Assessments Initiative Questionnaire (CAIQ), which is a set of Boolean yes/no

security-related questions (e.g., 'are all requirements and trust levels for customers' access defined and documented?') that can further help a user to assess security guarantees.

We close this section by highlighting some recent attempts towards incorporating security guarantees into SLAs, also known as secSLAs (e.g., [3, 20]). The key idea is that secSLAs should include information on the security controls implemented by the provider, their associated metrics (i.e., criteria and techniques for their evaluation), and the values guaranteed by the provider during service delivery. In this way, traditional approaches (e.g., approaches based on QoS) for assessing and selecting cloud plans could automatically take into account the security requirements of users as well as the security guarantees offered by cloud providers [7].

3 Requirements specification

The techniques illustrated in the previous section mainly deal with the problems of identifying attributes relevant for the evaluation of candidate plans or of developing techniques for the evaluation process. Orthogonally to these problems, there is also the need of allowing users to easily express their requirements to discriminate those plans that are suitable for outsourcing. The framework in [6] addresses this need by proposing a high-level and user-friendly language for expressing *requirements* and *preferences*. Requirements are hard constraints that a plan must satisfy to be acceptable for outsourcing. Preferences are soft constraints evaluated against acceptable plans (i.e., plans satisfying the requirements) and that can help in producing a rank among such acceptable plans: the higher the position of a plan in the ranking, the closer the plan to the needs of the user. The evaluation of requirements and preferences is executed by a broker, which verifies them against the characteristics of the plans, called *attributes* in [6], and returns to the user the computed plan ranking (Figure 3). Attributes might be metadata associated with the provider of a plan or, in general, any measurable property. We now illustrate more in details the specification language for requirements and preferences and the strategies for enforcing them. We will refer our examples to a set of attributes modeling, for each plan, the provider (`prov`), the geographical location of its servers (`loc`), the adopted encryption scheme (`encr`), the guaranteed availability (`avail`), the authority running penetration testing (`test`), the possessed security certification (`cert`), and the security auditing frequency (`aud`).

Requirements specification and enforcement. The building block of the requirements specification language is the *attribute term*. An attribute term t states that an attribute must assume a certain set of values (denoted $\texttt{attribute}(v_1, \ldots, v_n)$) or that, on the contrary, cannot assume a certain set of values (denoted $\neg\texttt{attribute}(v_1, \ldots, v_n)$) in its domain. For instance, attribute term '$t = \texttt{prov}(\text{Ghost}, \text{Mist}, \text{Cloudy})$' states that a plan must be offered by providers Ghost, Mist, or Cloudy. Starting from this building block, the proposed requirement specification language allows users to specify in a flexible way

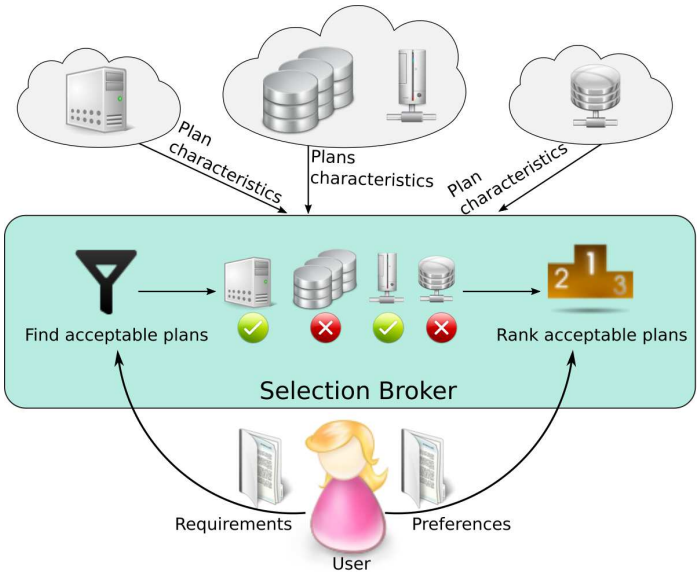

Fig. 3. Cloud plan selection and ranking with requirements and preferences [6]

a variety of requirements. The language supports the definition of the following requirements.

- *Base requirement.* It corresponds to an attribute term t, requiring that an attribute assumes/does not assume a certain set of values. For instance, a basic requirement of the form '$\texttt{prov}(\text{Ghost}, \text{Mist}, \text{Cloudy})$' states that a plan is considered acceptable only if it is offered by providers Ghost, Mist, or Cloudy.
- ANY *requirement.* It models alternatives among base requirements. For instance, a requirement of the form '$\text{ANY}(\{\texttt{loc}(\text{EU}), \texttt{cert}(\text{cert\_}\gamma)\})$' states that a plan is considered acceptable only if its servers are geographically located in the EU or if it has certification 'cert_γ'.
- ALL *requirement.* It represents sets of base requirements that must be jointly satisfied. For instance, '$\text{ALL}(\{\texttt{loc}(\text{EU}, \text{US}), \neg\texttt{encr}(\text{DES})\})$' states that a plan is considered acceptable only if servers are located in the EU or the US, and if the adopted encryption is not DES.
- IF–THEN *requirement.* It specifies that certain base requirements (those appearing in the THEN part) must be satisfied every time other base requirements (those appearing in the IF part) are also satisfied. For instance, '$\text{IF ALL}(\{\texttt{loc}(\text{US}), \texttt{encr}(\text{3DES}))$ THEN $\text{ANY}(\texttt{audit}(\text{3M}, \text{6M}), \texttt{cert}(\text{cert\_}\alpha))$' states that if a plan has servers in the US and encrypts with 3DES, then it must be audited for security every three or six months, or have certification 'cert_α'.

- FORBIDDEN *requirement*. It identifies forbidden configurations, that is, combinations of base requirements that cannot be all satisfied at the same time by an acceptable plan. For instance, 'FORBIDDEN($\{\neg$`loc`(EU), `test`(authC)$\}$)' states that a plan with servers not located in the EU and tested by authC is not acceptable.
- AT_LEAST *requirement*. It demands that at least n among a set of base requirements be satisfied. For instance, 'AT_LEAST$(2, \{$`loc`(EU), `encr`(AES), `prov`(Mist, Ghost)$\}$)' states that a plan is acceptable only if at least two among the conditions 'having servers within the EU', 'adopting AES encryption', and 'having Mist or Ghost as provider' are satisfied.
- AT_MOST *requirement*. It demands that at most n among a set of conditions be satisfied. For instance, 'AT_MOST$(2, \{$`prov`(Ghost), `avail`(M, MH), `encr`(3DES)$\}$)' states that a plan is acceptable only if at most two among the conditions 'being offered by provider Ghost', 'having a medium (M) or medium-high (MH) availability', and 'adopting 3DES encryption' are satisfied.

A plan is considered acceptable by a user iff it satisfies all her requirements. Given a set of requirements and a set of cloud plans, the approach in [6] checks whether the plans are acceptable using a Boolean interpretation of the requirements. For example, consider the plans in Figure 4(a) (abstractly represented as vectors with one element for each attribute reporting the value assumed by the attribute in the plan or symbol '—' if not specified) and the set $r_1, \ldots, r_{10}$ of requirements in Figure 4(b). It is easy to see that only plans P_1, P_2, and P_3 are acceptable, as P_4 does not satisfy requirements r_3, r_4, r_8, and r_{10}.

Preferences specification and enforcement. Like requirements, also preferences (used by the broker to rank acceptable plans) can be specified by the user, and the approach in [6] aims to support users with an intuitive specification model. In particular, we consider the following two levels of specifications for preferences:

- *attribute values*, to specify that certain values are more preferred than others (e.g., for attribute `encr`, a user might state that she prefers AES over 3DES); and
- *attributes*, to specify the importance that each attribute has for the user (e.g., a user interested in outsourcing mission-critical but non-sensitive data might state that attributes related to performance are more important than attributes related to security).

Preferences on attribute values are expressed as a total order relationship among sets of values that attributes can assume (i.e., the attribute domain is partitioned and preferences represent a total order relationship among partitions of values). For instance, if attribute `prov` can assume values Cloudy, Mist, and Ghost, a user might specify an ordering stating that Cloudy is preferred over Mist, which is in turn preferred over Ghost. Preferences on attributes are instead

	P_1	P_2	P_3	P_4	
`prov`	Mist	Mist	Mist	Cloudy	*cloud provider*
`loc`	US	EU	US	JP	*geographical location of servers*
`encr`	AES	AES	AES	DES	*adopted encryption*
`avail`	H	VH	VH	ML	*availability level*
`test`	authA	authA	authB	authC	*penetration test authority*
`cert`	cert_γ	cert_α	cert_γ	cert_γ	*security certification*
`aud`	—	—	—	3M	*security auditing frequency*

(a)

r_1 : `prov`(Ghost, Mist, Cloudy)
r_2 : ¬`avail`(VL, L)
r_3 : ALL({`loc`(EU, US), ¬`encr`(DES)})
r_4 : ANY({`test`(authA, authB), `cert`(cert_α, cert_β)})
r_5 : ANY({`loc`(EU), `cert`(cert_γ)})
r_6 : IF ALL({`loc`(US), `encr`(3DES)) THEN ANY(`audit`(3M, 6M), `cert`(cert_α))
r_7 : IF ALL(`test`(—)) THEN ANY(`cert`(cert_α))
r_8 : FORBIDDEN({¬`loc`(EU), `test`(authC)})
r_9 : AT_MOST(2, {`prov`(Ghost), `avail`(M, MH), `encr`(3DES)})
r_{10} : AT_LEAST(2, {`loc`(EU), `encr`(AES), `prov`(Mist, Ghost)})

(b)

Fig. 4. Abstract representation of cloud plans (a) and set of user requirements (b)

defined through a weight function that assigns a weight to each attribute. For instance, with reference to the example above, attributes related to performance can be assigned higher weights than attributes related to security. Figure 5 illustrates an example of preferences for the plans in Figure 4(a). Preferences on attribute values are graphically represented as a hierarchy among attribute values, with preferred elements appearing higher in the hierarchy. For each value, the figure also represents the relative position of the value in the ordering (with the most preferred value having preference 1, and the least preferred value having preference $1/k$, with k the number of partitions). Preferences on attributes are instead reported in round brackets on the right side of each attribute: in this example, all attributes have the same weight (1) except attribute `avail` (which has weight 10).

To rank plans based on preferences, the approach in [6] defines three possible strategies, including the intuitive Pareto-based ranking, and two distance-based rankings. According to the Pareto-based ranking, a plan P_i is preferred over a plan P_j if, for all attributes, its values are equally or more preferred than those in P_j and for at least one attribute, P_i has a more preferred value than P_j. For instance, Figure 6(a) illustrates the Pareto-based ranking computed over the plans in Figure 4(a), considering the preferences in Figure 5. As it is visible from this figure, P_1 dominates P_2 since they have the same value for `prov`, `encr`, `avail`, and `aud`, but P_1 has more preferred values for `loc`, `test`, and `cert`. On the contrary, P_2 and P_3 are not comparable. Distance-based rankings consider plans as points in an m-dimensional space (with m the number of attributes), located

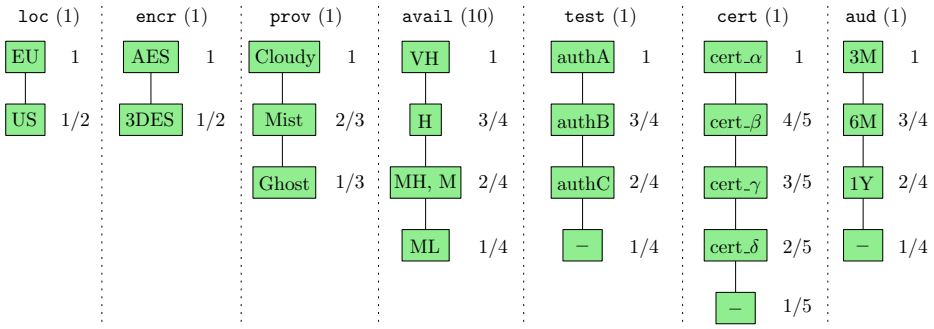

Fig. 5. User preferences for the plans in Figure 4(a)

through coordinates that are the relative positions assumed by their attribute values in the rankings induced by the preferences. For instance, with reference to the plans in Figure 4, plan P_1 has coordinates $[2/3, 1, 1, 1, 1, 1, 1/4]$ since, for example, it assumes value Mist for attribute `prov` which has a relative position of $2/3$ in the preferences in Figure 5. The ranking of cloud plans is then based on how distant each plan is from an *ideal plan* (i.e., a possibly non-existing plan that assumes, for each attribute, one of the top preferred values and has therefore coordinate equal to 1 for each attribute), with closer plans ranked higher. Distance can possibly be measured taking into account attribute weights. In the latter case, the relative position of each attribute value is multiplied by the weight of the corresponding attribute (i.e., attribute preferences are interpreted as scaling factors on the m-dimensional space). Figure 6(b) illustrates the distance-based rankings over the plans in Figure 4(a), considering the preferences in Figure 5. The ranking on the left does not consider preferences among attributes, while the one on the right takes attributes preferences into consideration. For each plan, the figure reports the scores assumed by attribute values, and used as coordinates in the m-dimensional space, and the distance (in boldface on the right-hand-side of each node) from the ideal plan.

4 Fuzzy logic for flexible requirements specification

The approaches illustrated in the previous sections mainly operate on crisp values assumed by generic attributes of cloud plans. However, reasoning directly over crisp, and possibly low-level, characteristics of cloud plans implicitly assumes that users are familiar with technical details of the cloud environment to differentiate, for example, the attractiveness of a plan offering an availability of 99.99% from that of a plan offering 99.98%. This assumption might be limiting in some real-world scenarios, for two main reasons. First, users might not possess technical skills allowing them to fully understand the low-level characteristics of a cloud plan, and hence to formulate complete and/or sound requirements precisely capturing their needs. Second, operating on crisp values inevitably intro-

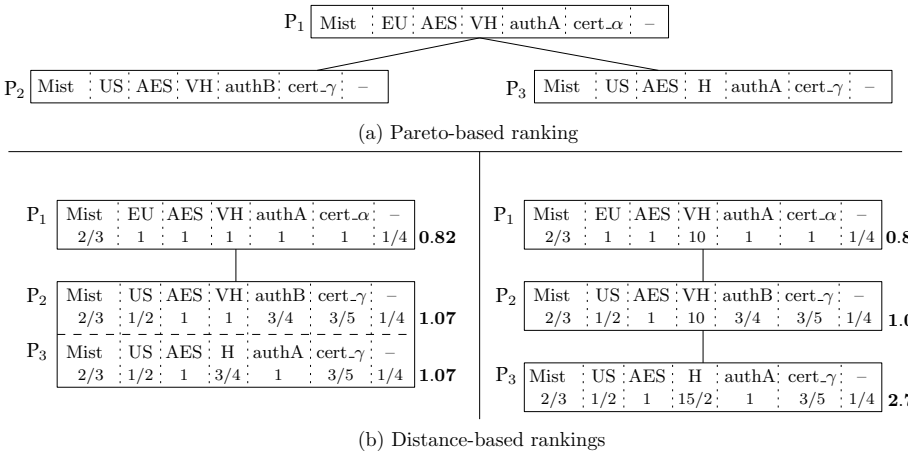

Fig. 6. Rankings of plans P_1, P_2, P_3 in Figure 4(a) that satisfy the requirements in Figure 4(b) and considering the preferences in Figure 5

duces sharp boundaries between 'good' and 'bad' values, while human reasoning is typically more flexible and good and bad values might slightly overlap.

To overcome these limitations, a possible solution relies on the adoption of fuzzy logic [7, 12]. In fact, by permitting to reason with linguistic values (such as 'high', 'low', 'good', and 'bad') and imprecise information (and providing the mathematical foundation for approximate reasoning, mapping linguistic/imprecise information to the actual characteristics of cloud plans), fuzzy logic can help users in formulating requirements and preferences in a way that is more similar to human reasoning, which entails intrinsic imprecision and vagueness. Fuzzy logic can then allow users to define their application needs in a flexible way, capturing natural linguistic expressions, when users are not specialists in information systems and technologies and when requirements are not easily definable.

In particular, the proposal in [12] uses fuzzy logic to support the definition of both user requirements in terms of *fuzzy parameters* and *fuzzy concepts*, as well as the importance of (crisp) requirements.

Fuzzy parameters. Fuzzy parameters permit to define requirements when users are unable to determine a specific value of a characteristic of the cloud environment, but they are fully conscious of the required size of the considered characteristic and are linguistically able to describe it (e.g., with adjectives of periphrases). To illustrate, suppose that a provider allows users to choose among several key lengths for encrypting data at rest or in transit, and consider a non technically skilled user who wishes to outsource her medical data. Being her data sensitive, the user wants confidentiality to be guaranteed and, for this reason, she would like to use a long encryption key. If the user does not have a precise idea of the needed key length, she may prefer to simply state that '`key_length` should

Fuzzy label	Range
very short	1-32 bit
short	16-128 bit
medium	64-256 bit
long	128-1024 bit
very long	512-2048 bit

(a)

Fuzzy concept	Parameters
high data security	encryption: AES
	min. key length: 256 bit
	HMAC: SHA-512
	hash key length: 512 bit

(b)

Fig. 7. An example of fuzzy specification of key length parameter (a) and of data security concept (b)

be long', accepting a conventional definition of 'long' key as a fuzzy range of values. A common vocabulary about the meaning of linguistic expressions must be shared between the user and the provider to understand and satisfy user requirements. Figure 7(a) illustrates an example of fuzzy vocabulary for the key length property. The separation between ranges of values for key length is not crisp, but ranges may overlap. Note that, besides helping users in formulating requirements, such a fuzzy specification of requirements allows cloud providers to manage with higher elasticity their resources. Indeed, fuzzy specification enables users to express flexible requirements that cloud providers can satisfy without leaving resources unused when applications do not explicitly demand for them. Consider, as an example, two applications expressing requirements on storage space and a cloud provider with 1.9TB of free space. The provider could not accommodate two applications requiring 1TB of storage space, while it could manage them if requesting large storage space, where large is between 0.7TB and 1TB and the first application actually uses 0.8TB and the second one uses 0.95TB. The definition of fuzzy parameters enables for better resource allocation, with higher quality of service at lower costs for both the provider and the users.

Fuzzy concepts. While supporting users in requirements formulation, fuzzy parameters can still require some technological competence to users (with reference to the example above, a user formulating a fuzzy requirement over the key length parameter should still know that the length of an encryption key typically impacts the offered protection). Fuzzy logic can also provide a further level of support, by operating on an abstract level more easily accessible also to non-skilled users. To this end, fuzzy logic can operate on *fuzzy concepts*, that is, high level features that do not directly correspond to a cloud characteristic or parameter, but map on an appropriate combination of them. In this context, fuzzy logic can provide the mathematical foundation for merging real characteristics and metrics, translating the linguistic high-level description given by the user. To illustrate, consider the example above and suppose that the user is agnostic about the security provided by different encryption algorithms and key lengths. If the user is still wishing to protect her medical data upon outsourcing, she may simply prefer to request 'high data security' instead of specifying which

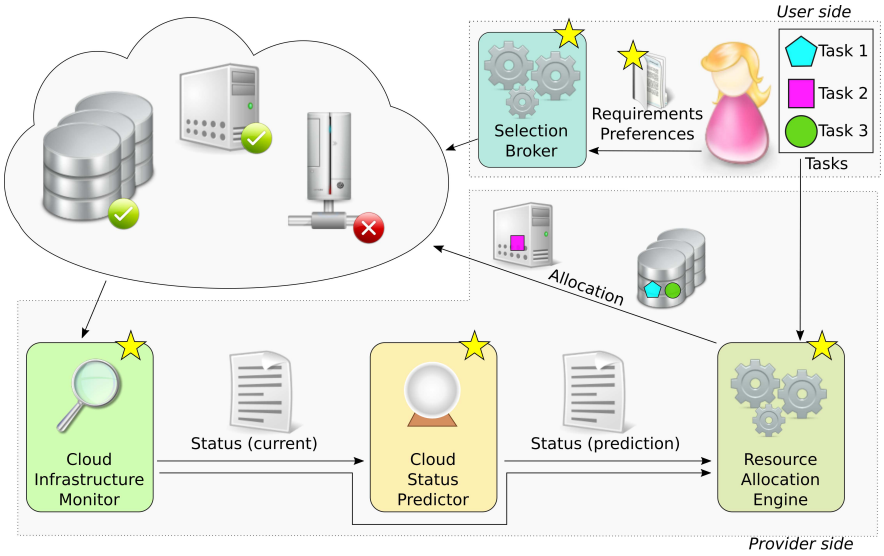

Fig. 8. Possible applications of fuzzy logic in cloud selection and management

algorithm or key length is appropriate (Figure 7(b)). Such high-level requirement can then be formalized and processed through fuzzy logic, translating it into an equivalent combination of parameter values to be guaranteed by the provider.

Weighting crisp requirements. Fuzzy logic might also be used to assign a weight, or importance level, to a set of crisp requirements specified by the user (e.g., like those illustrated in Section 3). Weighting requirements becomes more relevant when, for any reason, not all of them can be satisfied at the same time (e.g., when the response time grows above the requested threshold in case of a burst of incoming requests, or heavy workload). If requirements do not have the same relevance to the user, fuzzy logic might be employed to specify the importance of each requirement in such a way to discriminate between critical requirements (whose satisfaction must always be guaranteed) and secondary ones (whose satisfaction is important, but less than that of critical ones). For instance, when outsourcing a mission-critical application that needs to be up and running 24/7 with no delays, the user might specify that the availability requirement has 'high importance', while storage requirement has 'medium importance' and user interface and interaction have 'low importance'.

Fuzzy parameters, fuzzy concepts, and fuzzy importance of crisp requirements can then be transformed in a format that can be processed in a homogeneous way with other crisp requirements having a crisp weight, to take all of them into account in a comprehensive strategy.

We close this section observing that, besides being applicable at the user side for specifying requirements, fuzzy logic can prove beneficial also at the

provider side, that is, in the low-level management of the cloud resources (e.g., CPU or virtual machine instances allocation) [1, 5, 11–13, 21, 22, 25, 31]. Figure 8 graphically illustrates a high-level representation of a cloud management system, including a user (with requirements and preferences over the characteristics of cloud plans), and a set of provider-side technological components that manage the overall service provision. We graphically highlight the possible adoption of fuzzy logic with a star on the corresponding component/interaction among parties. In particular, by making available flexible reasoning possibly with imprecise/partial information, fuzzy logic can be used at the provider side to: *i)* continuously monitor the cloud infrastructure (*cloud infrastructure monitor* in the figure) to identify and characterize the current status of the cloud environment; *ii)* predict the future status of the infrastructure (*cloud status predictor* in the figure), for example, to forecast peaks in incoming requests; and *iii)* flexibly allocate resources to the tasks required by the user applications (*resource allocation engine* in the figure), for example, to scale up or down allocated resources when higher or lower demands are forecasted or observed.

5 Conclusions

Selecting the right cloud plan when outsourcing data and applications to the cloud is a key issue for ensuring a satisfying experience for users. The problems related to cloud plan selection are challenging and diverse, and the scientific community has recently addressed them by proposing models and techniques that support users in assessing a set of cloud plans to select the right one. In this chapter, we have illustrated some of the existing techniques for determining attributes for evaluating cloud plans, for practically evaluating users' requirements and desiderata to assess a set of candidate plans, and for supporting users in the specification of their requirements and preferences. We have also highlighted how fuzzy logic can be beneficial in cloud plan selection.

Acknowledgments

This work was supported in part by the EC within the H2020 under grant agreement 644579 (ESCUDO-CLOUD), and within the FP7 under grant agreement 312797 (ABC4EU).

References

1. Anglano, C., Canonico, M., Guazzone, M.: FC2Q: Exploiting fuzzy control in server consolidation for cloud applications with SLA constraints. Concurrency and Computation: Practice and Experience 22(6), 4491–4514 (2014)
2. Arman, A., Foresti, S., Livraga, G., Samarati, P.: A consensus-based approach for selecting cloud plans. In: Proc. of IEEE RTSI 2016. Bologna, Italy (September 2016)

3. Casola, V., De Benedictis, A., Eraşcu, M., Modic, J., Rak, M.: Automatically enforcing security SLAs in the cloud. IEEE Transactions on Services Computing (TSC) 10(5), 741–755 (2017)
4. Cloud Security Alliance: Cloud Control Matrix v3.0.1. https://cloudsecurityalliance.org/research/ccm/, online, accessed on June 05, 2018
5. Dastjerdi, A.V., Buyya, R.: Compatibility-aware cloud service composition under fuzzy preferences of users. IEEE Transactions on Computers (TCC) 2(1), 1–13 (2014)
6. De Capitani di Vimercati, S., Foresti, S., Livraga, G., Piuri, V., Samarati, P.: Supporting user requirements and preferences in cloud plan selection. IEEE Transactions on Services Computing (TSC) (2017)
7. De Capitani di Vimercati, S., Foresti, S., Livraga, G., Samarati, P.: Supporting users in data outsourcing and protection in the cloud. In: Helfert, M., Ferguson, D., Munoz, V., Cardoso, J. (eds.) International Conference on Cloud Computing and Services Science. Springer (2017)
8. De Capitani di Vimercati, S., Livraga, G., Piuri, V.: Application requirements with preferences in cloud-based information processing. In: Proc. of IEEE RTSI 2016. Bologna, Italy (September 2016)
9. De Capitani di Vimercati, S., Livraga, G., Piuri, V., Samarati, P., Soares, G.: Supporting application requirements in cloud-based IoT information processing. In: Proc. of IoTBD 2016. Rome, Italy (April 2016)
10. Ding, S., Wang, Z., Wu, D., Olson, D.L.: Utilizing customer satisfaction in ranking prediction for personalized cloud service selection. Decision Support Systems 93, 1–10 (2017)
11. Esposito, C., Ficco, M., Palmieri, F., Castiglione, A.: Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory. IEEE Transactions on Computers (TCC) 65(8), 2348–2362 (2016)
12. Foresti, S., Piuri, V., Soares, G.: On the use of fuzzy logic in dependable cloud management. In: Proc. of IEEE CNS 2015. Florence, Italy (September 2015)
13. Frey, S., Lüthje, C., Reich, C., Clarke, N.: Cloud QoS scaling by fuzzy logic. In: Proc. of IEEE IC2E 2014. Boston, MA, USA (March 2014)
14. Garg, S.K., Versteeg, S., Buyya, R.: A framework for ranking of cloud computing services. Future Generation Computer Systems 29(4), 1012–1023 (2013)
15. Ghosh, N., Ghosh, S.K., Das, S.K.: SelCSP: A framework to facilitate selection of cloud service providers. IEEE Transactions on Computers (TCC) 3(1), 66–79 (2015)
16. Jatoth, C., Gangadharan, G., Fiore, U., Buyya, R.: SELCLOUD: a hybrid multi-criteria decision-making model for selection of cloud services. Soft Computing pp. 1–15 (2018)
17. Jhawar, R., Piuri, V., Samarati, P.: Supporting security requirements for resource management in cloud computing. In: Proc. of IEEE CSE 2012. Paphos, Cyprus (December 2012)
18. Li, A., Yang, X., Kandula, S., Zhang, M.: CloudCmp: Comparing public cloud providers. In: Proc. of ACM IMC 2010. Melbourne, Australia (November 2010)
19. Li, J., Squicciarini, A.C., Lin, D., Sundareswaran, S., Jia, C.: MMB^{cloud}-tree: Authenticated index for verifiable cloud service selection. IEEE Transactions on Dependable and Secure Computing (TDSC) 14(2), 185–198 (2017)
20. Luna, J., Suri, N., Iorga, M., Karmel, A.: Leveraging the potential of cloud security service-level agreements through standards. IEEE Cloud Computing 2(3), 32–40 (2015)

21. Patiniotakis, I., Rizou, S., Verginadis, Y., Mentzas, G.: Managing imprecise criteria in cloud service ranking with a fuzzy multi-criteria decision making method. In: Proc. of ESOCC 2013. Malaga, Spain (September 2013)
22. Patiniotakis, I., Verginadis, Y., Mentzas, G.: PuLSaR: Preference-based cloud service selection for cloud service brokers. Journal of Internet Services and Applications 6(26), 1–14 (2015)
23. Pawluk, P., Simmons, B., Smit, M., Litoiu, M., Mankovski, S.: Introducing STRATOS: A cloud broker service. In: Proc. of IEEE CLOUD 2012. Honolulu, HI, USA (June 2012)
24. Qu, L., Wang, Y., Orgun, M.A., Liu, L., Liu, H., Bouguettaya, A.: CCCloud: Context-aware and credible cloud service selection based on subjective assessment and objective assessment. IEEE Transactions on Services Computing (TSC) 8(3), 369–383 (2015)
25. Rao, J., Wei, Y., Gong, J., Xu, C.Z.: DynaQoS: Model-free self-tuning fuzzy control of virtualized resources for QoS provisioning. In: Proc. of IEEE IWQoS 2011. San Jose, CA, USA (June 2011)
26. Rehman, Z., Hussain, O., Hussain, F.: IaaS cloud selection using MCDM methods. In: Proc. of IEEE ICEBE 2012. Hangzhou, China (September 2012)
27. Ruiz-Alvarez, A., Humphrey, M.: An automated approach to cloud storage service selection. In: Proc. of ACM ScienceCloud 2011. San Jose, CA, USA (June 2011)
28. Samarati, P.: Data security and privacy in the cloud. In: Proc. of 10th International Conference on Information Security Practice and Experience (ISPEC 2014). Fuzhou, China (May 2014)
29. Samarati, P., De Capitani di Vimercati, S.: Cloud security: Issues and concerns. In: Murugesan, S., Bojanova, I. (eds.) Encyclopedia on Cloud Computing. Wiley (2018)
30. Samreen, F., Elkhatib, Y., Rowe, M., Blair, G.S.: Daleel: Simplifying cloud instance selection using machine learning. In: Proc. of IEEE/IFIP NOMS 2016. Istanbul, Turkey (April 2016)
31. Sun, L., Ma, J., Zhang, Y., Dong, H., Hussain, F.K.: Cloud-FuSeR: Fuzzy ontology and MCDM based cloud service selection. Future Generation Computer Systems 57, 42–55 (2016)
32. Sundareswaran, S., Squicciarini, A., Lin, D.: A brokerage-based approach for cloud service selection. In: Proc. of IEEE CLOUD 2012. Honolulu, HI, USA (June 2012)
33. Tang, M., Dai, X., Liu, J., Chen, J.: Towards a trust evaluation middleware for cloud service selection. Future Generation Computer Systems 74, 302–312 (2017)
34. Zheng, Z., Wu, X., Zhang, Y., Lyu, M.R., Wang, J.: QoS ranking prediction for cloud services. IEEE Transactions on Parallel and Distributed Systems (TPDS) 24(6), 1213–1222 (2013)