Fuzzy-Engineered Multi-Cloud Resource Brokering for Data-intensive Applications

Ashish Pandey, Prasad Calyam, Zhen Lyu, Trupti Joshi
University of Missouri-Columbia, USA
{*apfd6, zl7w2*}*@mail.missouri.edu*; *calyamp@missouri.edu joshitr@health.missouri.edu*

***Abstract*—Multi-cloud resource brokering is becoming a critical requirement for applications that require high scale, diversity, and resilience. Applications demand timely selection of distributed data storage and computation platforms that span local private cloud resources as well as resources from multiple cloud service providers (CSPs). The distinct capabilities and policies, as well as performance/cost of the cloud services, are amongst the prime factors for CSP selection. However, application owners who need suitable cyber resources in community/public clouds, often have preliminary knowledge and preferences of certain CSPs. They also lack expert guidance to handle the problem of overwhelming resource choice from CSPs, and optimization to compensate for service dynamics. In this paper, we address this challenge of optimal resource selection while also leveraging limited user's expertise and preferences towards CSPs through multi-level fuzzy logic modeling based on convoluted factors of performance, agility, cost, and security. We evaluate the efficiency of our fuzzy-engineered resource brokering in improving allocation of resources as well as user satisfiability by using case studies and independent validations of CSPs evaluation.**

***Index Terms*—Multi-cloud resource recommendation, Performance optimization, Custom cloud templates, Fuzzy logic**

I. Introduction

Application developers working with data-intensive applications such as social media platforms, income tax filing and other mass media platforms often require high-scale, diverse and fault-tolerant high performance computing, networking and storage resources. Such a requirement in many cases today cannot be fulfilled by using private servers in a data center due to e.g., cost, power, heating/cooling and personnel expertise issues. The private servers with capacity limitations would need exorbitant amount of time to deliver satisfactory application performance. This necessitates applications to leverage advanced and scalable resources such as high end processors, RAM and GPUs, from community/public cloud service providers (CSPs) [1][2]. However, limited and biased expertise among users in composing and deploying suitable multi-cloud architectures may cause delays in deployment, or in some cases cause bottlenecks (e.g., in terms of time, cost) in the deployment or functions after deployment.

To mitigate the problem of ill-advised resource allocation, there is a need for systematic studies to identify and configure resources based on key factors such as performance, agility, cost and security (PACS) offered from the CSPs [1]. Since selection and configuration of multi-cloud resources for modern applications requires handling PACS factors, the multi-cloud resource brokering involves a multi-dimensional optimization problem. Apart from these functional and objective factors, the users often gain expertise towards certain CSPs creating inherent biases in CSP selection depending on the functional requirements of PACS or business and geographical constraints to name a few. Moreover PACS factors are subjective factors which can have varying metric for evaluation for different cloud service users (CSUs). For example, performance can be evaluated based on sub-factors of availability, reliability, response time and throughput. The measurement of these sub-factors influence the evaluation of PACS factors and ultimately the CSP selection.

Resource configuration and management service suites such as AWS OpsWorks [3] give insight about trade offs among its services in context of PACS criteria, but these tools generally focus on a brokering a single cloud platform pool of resources. Moreover, the tools do not consider subjective factors arising from users' bias toward certain cloud platforms or perceived performance of the cloud resource by users, which also needs to be considered in the multi-cloud resource brokering. The subjective experience of users can also fluctuate depending on the quality of service (QoS), type of applications, capacity load on the servers, and location of the servers to name a few, and thus can not be quantified. To gain meaningful insights of the bias towards cloud providers which is affected by PACS criteria, fuzzy logic can potentially be promising [4]. Fuzzy logic theory gives tools and methodologies to study uncertainty in a system or a situation and provides flexibility in reasoning. The idea behind fuzzy logic is to imitate human behavior and logical reasoning for deducing conclusions for vague problems in a non-linearly weighted manner [5].

In this paper, we propose a multi-level fuzzy logic controller viz., *OnTimeFLC* for multi-cloud resource brokering of data-intensive applications considering users perspectives. Figure 1 shows core components and control flow for creating the multi-level FLC in order to capture, mimic and utilize application developers' preferences/expertise of CSPs resources. We propose a model that contains two levels of fuzzy logic inference engine to evaluate CSPs for: (a) evaluation of PACS factors from their respective sub-factors, and (b) CSP evaluation from their respective PACS factors. We formulate the multi-cloud resource selection problem from a user's perspective when they desire optimal solutions for deploying their applications. Our optimizer prescriptively recommends an intelligent set of customized cloud template solutions optimized based on PACS criteria over multi-cloud resources. Our OnTimeFLC implementation involves integration of our optimizer based on integer linear programming (ILP) aided by the multi-

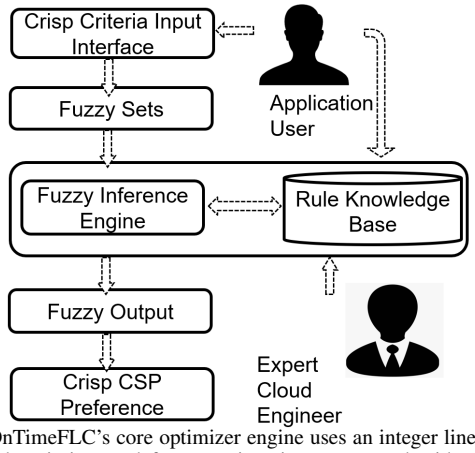

Fig. 1: OnTimeFLC's core optimizer engine uses an integer linear programming based optimizer and fuzzy engineering augmented with a knowledge base from different cloud providers: (i) Amazon Web Services (AWS) [6] - public cloud, (ii) GENI - community cloud [7], (iii) MU Data Center - private cloud [8].

level FLC model for providing users with three sets of cloud template solutions classified as: (i) *Red* (low-cost), (ii) *Green* (high-performance), and (iii) *Gold* (optimal cost-performance) templates [9].

For evaluation of our OnTimeFLC implementation, we conduct experiments with a catalog of compute and data intensive bioinformatics applications featuring varying type and size of datasets. We consider three CSP resources (i.e., Amazon Web Services, GENI and MU) featuring more than 300 different machine configuration instances in our simulations and experiments. We compare the improvement in performance of application execution through resource brokering 'with' and 'without' our proposed fuzzy logic controller. We also evaluate accuracy of fuzzy modeling through use cases and independent validations where we characterize PACS metrics for three above mentioned CSPs based on CSUs preferences.

The remainder of this paper is organized as follows: In Section II, we describe related works. Section III presents the multi-cloud resource selection problem background and our solution architecture description. Section IV presents details of our OnTimeFLC resource broker components and their implementation. Section V discusses the testbed & performance evaluation. Section VI concludes the paper.

II. Related Work

Fuzzy logic in user cloud selection: There have been comprehensive studies on the behavior and utilities of fuzzy models in simulating decision making [10] [11] which can be simulated for cloud service selection decision. Specifically, fuzzy logic can be used to complement multi-cloud resource brokering methods that take into account quantitative user resource specifications [1]. These methods can leverage fuzzy logic modeling to consider user's expertise and biases in cloud selection using the integer linear optimization approaches. Since fuzzy logic can be used in decision making, it has also been used by researchers in modeling optimal scheduling of resources on cloud infrastructure on data centers. In [10], the researchers have evaluated different characteristic of fuzzy models and their accuracy in decision making. The study aimed to compare fuzzy systems with different configurations to predict the surface temperature of broiler chickens subjected to different intensities and duration of thermal challenges in the second week of life. The study concludes that while developing fuzzy systems, different configurations must be compared, and the system with smaller simulation errors should be selected.

Fuzzy logic have also been used in the domain of quantifying resource selection. Exemplar efforts in characterizing cloud resources can be seen in [4]. The authors emphasize that there is need of measuring and evaluating cloud performance to help the users in making their decisions. The researcher primarily aimed to develop a model to evaluate the performance of cloud based on factors such as workload, storage, hypervisor, and network devices. Their proposed model clarifies how the infrastructure and applications on a cloud platform can play an important role in the application performance delivery. However the study is focused only for performance benchmarking for a single cloud platform.

A similar work [12] aims to facilitate decision making process in order to determine the choice of service provider for a particular process by using a rule-based fuzzy logic technique and deep learning. The decision is based on parameters such as priority, age and execution time required for the process. However, the approach requires a very large knowledge base of data with details about the tasks to be processed. In a closely related work [11], authors present the state-of-the-art approaches and their important features in fuzzy logic based cloud computing. The work focuses on the use of fuzzy logic in cloud computing with different membership functions and with different type of defuzzification methods. Fuzzy logic have also been used in improving resources allocation strategies for scientific applications [13] where authors propose a novel elasticity controller for autonomic resource provisioning which is a combination of fuzzy logic control and autonomic computing. Results show that their proposed approach minimizes execution time and resource utilization by the applications.

Fuzzy logic in Cloud platforms: Many researchers have attempted in using fuzzy logic for scheduling of resources [14] [15] [16]. Amin et. al. [14] have presented and evaluated a new scheduling algorithm that is an efficient technique for scheduling virtual machines between data centers using fuzzy logic. A similar work [15] focuses on allocation and scheduling of resources as well as improving the reliability of cloud computing. The fuzzy model is based on the governing rules which consider factors of cost, trust, length of processes and priority. Further, the authors showed that their output resulted in lower cost and increased reliability of cloud resources compared to the scheduling techniques such as FIFO (first in first out) and Min-Max (min requirement task first). Toosi et. al. [17] extended fuzzy logic into a load balancing algorithm. Their approach helps cloud providers with multiple geo-distributed data centers in a region by evaluating the temporal variations in on-site power and grid power price. It then optimizes by routing the demand to a suitable data center in order to reduce cost and improve energy utiliza-

tion. A similar work [16] proposes a multi-objective best-fit-decreasing (BFD) solution to the virtual machine reallocation problem. The authors consider a multi-objective formulation accounting for power costs and resource utilization. Although the methodologies followed in above works give better insights of the research domain and utility, the users experience with cloud service providers have not been considered in decision making progress of resource selection as we do in our work in this paper.

A recent work by Rizvi et al. [18] sought to evaluate security of CSPs through a fuzzy inference system. The authors used several sub-factors modeled with a fuzzy inference model for measuring security readiness of cloud providers in cloud service users' (CSUs) perspective. Our work is inspired by this prior effort and we further characterize cloud platforms based on performance, agility, cost and security (PACS) factors from a user's perspective of non-functional requirements. Our proposed model is unique compared to other related works because we consider the input from CSUs and their convoluted definition of PACS to then synthesize the information into a quantitative form, and finally evaluate cloud service provider selection. We further use the results to create optimal cloud templates customized to user preferences through a novel OnTimeFLC's optimizer augmented with a fuzzy logic model.

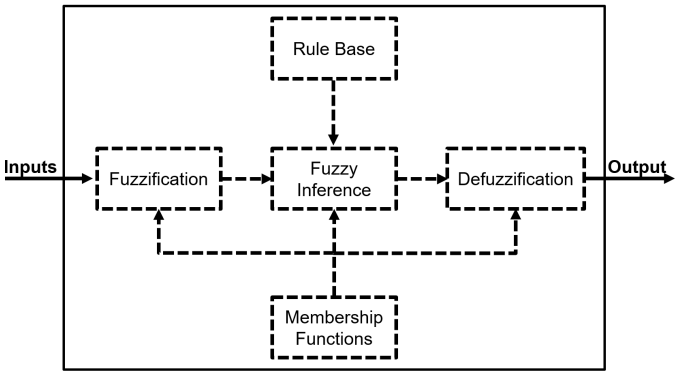

Fig. 2: A standard fuzzy inference system comprising of: (a) Input/Output variables, (b) Rule Base, (c) Inference Engine and, (d) Membership functions for variables.

III. Fuzzy-Engineering Brokering Solution Overview

To ease the process of multi-cloud resource brokering for non-expert cloud users, we divide our OnTimeFLC into four steps: (i) Collection, (ii) Composition, (iii) Consumption, and (iv) Fuzzy Engineering.

A. Collection

1) Fuzzy Interface System (FIS)

For collecting user resource requirements, we integrate a Fuzzy Interface System (FIS) in the KBCommons web portal [19][20][21]. The KBCommons portal is a science gateway where multiple scientific workflows are deployed by bioinformatics researchers/educators. We integrated our OnTimeFLC middleware in the KBCommons portal to collect user specific resource constraints and criteria to execute workflows as shown in Figure 4. The middleware presents users with a set of questions (e.g., number of vCPU/RAM required) organized into functional groups such as: storage, networking, computation, and software requirements. Collection process also requires users to identify themselves as expert/non-expert cyber users. Further, the FIS is designed to collect and understand users experience in using CSPs based on PACS and their sub-factors. The user are provided with questionnaires to rate their experiences towards different CSP services on a scale (1-100). The users are also given an option to create rules to quantify their experiences. The format of rule creation is governed by 'If', 'Then' and abbreviations shown in Table IV.

B. Composition

1) Fuzzy Inference Engine

The user inputs from FIS are evaluated using a fuzzy inference engine. The inference engine evaluates user inputs with their corresponding membership function against a set of pre-defined rule knowledge base as shown in Figure 2. The fuzzy inference system outputs the expected weights towards different CSPs in probability. These biases are further used to influence cloud solution template composition through OnTimeFLC's core integer linear programming (ILP) based optimization engine. Section IV contains detailed explanation of the multi-level inference engine. Below is a final sample output from the fuzzy system.

```
{ AWS : 0.50
  GENI : 0.35
  MU : 0.15
}
```

Listing 1: Sample fuzzy output from fuzzy inference model.

2) Template Composition

Once the user specified requirements for the application workflow are collected, they are input to OnTimeFLC framework to create template solutions forming a catalog similar to the work in Antequera et. al. [9]. Each template solution in the catalog is formatted as a JSON object comprising of a set of distinct machine node instances. Each node instance represents a distinct machine configuration in terms of the available processing units (e.g., CPUs, memory, network bandwidth). For example, $a1.medium$ is one of the machine instances provided from AWS. A sample representation of template is provided in Listing 2, wherein three distinct type of AWS intances are used to compose the template.

```
[{csp:AWS,name:t3.nano,count:2},
{csp:AWS,name:t3.micro,count:1},
{csp:AWS,name:t3.small,count:1}]
```

Listing 2: Sample template format consisting of three type of instance machines from AWS.

C. Consumption

Cloud templates generated by OnTimeFLC are selected and automatically deployed on cloud resources. Figure 4 illustrates the detailed steps of the template consumption and monitoring that we have designed and implemented for a custom scientific

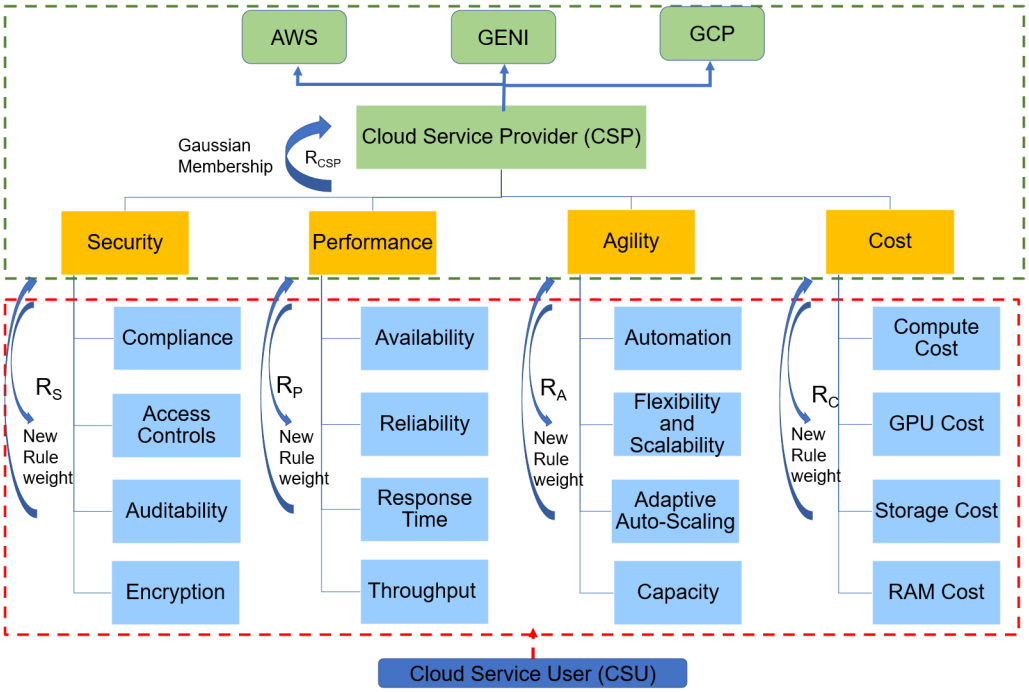

Fig. 3: Multi-Level fuzzy inference model with: (i) Red-box representing base fuzzy inference model for measuring PACS from sub-factors, and (ii) Green box representing fuzzy inference model for evaluating CSPs from PACS factors.

workflow on KBCommons. These workflows are based on the Pegasus workflow management system [22] and HTCondor job scheduler [23]. Pegasus and HTCondor are configured to run the application workflow on the deployed cloud resources recommended by the cloud template. Pegasus breaks the workflow tasks into sub-tasks mapped into a directed acyclic graph with edges representing dependencies, which are further scheduled on different cloud instances by HTCondor scheduler and pyGlidein glidein service [24].

Figure 4 displays all the components of Collection, Composition, Consumption and Fuzzy Engineering (explained in detail in Section IV) in synergy to create the end-to-end multi-cloud resource brokering solution. In addition to these four primary components, Cyverse [25][26] data portal is also integrated within the service management to enable users to store their raw data, which is staged in real-time to the OnTimeFLC recommended CSPs for processing with help of the 'iCommand' tool from Cyverse [25][26] and Pegasus [22] APIs.

IV. Fuzzy Engineering in OnTimeFLC Optimization

Selection of machines from a distributed set of diverse cloud instances under specific constraints is an NP-hard problem. To effectively formulate the problem, we use a CPLEX [27] optimizer to create the relevant optimization model. Our modeling objective is to reduce the effective cost of the template solution for specific user requirements by suitably considering constraints based on the threshold and user specification of required resources. We integrate the results from fuzzy model into the optimization model by effectively manipulating cost of instances so that the probability of selection instances from fuzzy model favored CSPs are increased. We discus below the key PACS and their sub-factors effecting CSP evaluation in the context of CSUs.

A. PACS: Key Criteria Optimizations

1) Performance: Categorization of Templates

OnTimeFLC is designed to produce multiple template solutions for different performance requirements. This categorization effectively provides users with multiple choices to select template solutions that have varying allocation of resources for better performance. ***Red Solution:*** Strict user defined resource constraints are considered. This is the most cost-effective optimal solution. The template is the closest match to the user resource specification with minimal over provisioning. Potential resource provisioning: {{*cpu:2, ram:4Gb, network: 200Mbps, storage: 15Gb*} & {*cpu:2, ram:4Gb, network: 200Mbps, storage: 15Gb*}}. ***Green Solution***: All user defined resources are amplified in step sizes up to a user defined threshold limit. Each step gives an amplified resource constraint which results in a corresponding over-provisioned solution. Maximum potential resource provisioning: {*cpu:**5**, ram:**10Gb**, network: **250Mbps**, storage: **37Gb***}. ***Gold Solution:*** It is similar to a green solution, but less over provisioning of resources is applied as per the user preferences. To evaluate performance for CSPs by the CSUs we have considered four sub-factors namely: *(i) Availability, (ii) Reliability, (iii) Response time, (iv) Throughput* as shown in Figure. 3.

2) Agility

To ensure agility of the template solutions, we use a parameter w^p that includes a weight on the cost of instances of the CSPs. The w^p will be a number between 1 to 10, and represents a global view of services and features provided from individual CSPs. A smaller w^p suggests fewer number

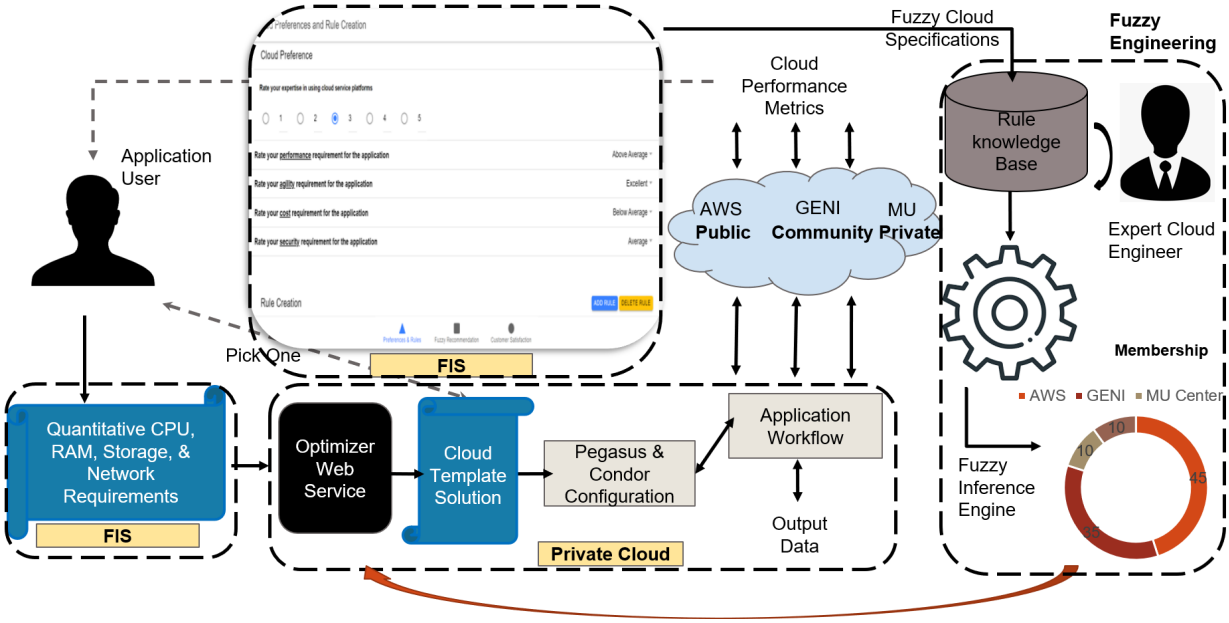

Fig. 4: Brokering lifecycle comprising of: (a) Collection - Fuzzy Interface System (FIS) deployed in KBCommons portal [19] to collect user specification; (b) Composition - Templates are composed and classified by optimizer into Red, Green, Gold templates which are presented in the FIS for selection; (c) Consumption: Pegasus, HTCondor and workflows are configured in a GENI [7] node machine, and HTCondor as per cloud template schedules tasks in resources of AWS, GENI or MU, and (d) Fuzzy Engineering - Rules and ratings on PACS criteria are collected and inferred, the results are input to optimizer.

of services from a CSP. The optimizer will also track the increase in cost to agility ratio of inclusion of instances from a CSP as per Equation 2. Weights w^p for agility of cloud providers can be assessed by the number of agile services provided from the provider (f^p) against a combined set of all agile services offered from all providers assessed by the user (F) as formulated in Equation 9, and normalized between 1 to 10. User discretion is used to identify agile services from CSPs as users will have varying standard of agility expectations. We use Cloudarado [28] as a reference data source to identify a set of agile services.

$$AgilityFactor(w^p) = \frac{f^p}{F} \tag{1}$$

To evaluate agility for CSPs by the CSUs we have considered four sub-factors namely: *(i) Automation, (ii) Flexibility and scalability, (iii) Adaptive-Scaling, and (iv) Capacity* as shown in Figure 3.

3) Cost: Optimization

We customize the cloud resource information to the optimizer in the JSON format detailed in Listing 3, based on real world data we extracted from more than 300 physical instance types from different CSPs that are candidates for optimization. These instances were considered because they are available on a pay-as-you-go basis from the corresponding CSPs. However, more instances can be added to this knowledge base and used by the optimizer. Given that the optimizer does not differentiate between these instance objects, the recommended template solutions could be a mix of resources from multiple CSPs. To evaluate agility for CSPs by the CSUs, we have considered four sub-factors namely: *(i) Compute cost, (ii) RAM cost, (iii) GPU cost, and (iv) Storage cost* as shown in Figure 3.

```
[{ "csp":"AWS",
    "OS":"LINUX",
    "name":"a1.medium",
    "vCPU":"1",
    "ram":"2",
    "price":"0.0255",
    "network":"10",
    "clock":"2.3",
    },{...} ]
```

Listing 3: Sample of JSON formatted machine instances provided to our optimizer engine.

To ensure hardware granularities provided as instances from CSPs, the optimization problem is formulated as an integer linear programming (ILP) model, which is convex and guarantees the best possible solution. Equation 2 ensures the objective to minimize the cost of template solution creation. The number of instances (x^p_i) are calculated subject to the constraints deployed in the model referenced in Equation 3. Constraints are created on instances such that the resources contributed from the deployed instances should satisfy user-specified requirements (S_t). Equation 4 adds constraints on instances that belong to incompatible CSPs using Δ_{p1p2}.

The parameters, as summarized in definitions Table I, are inputs for our model described with Equations 2,3, and 4.

$$minimize \sum_{p=1}^{P} \sum_{i=1}^{I^p} \frac{c_i^p}{w_i^p} \cdot x_i^p \cdot m^p \tag{2}$$

$$\text{subject to: } \sum_{p=1}^{P} \sum_{i=1}^{I^p} R_{it}^p \cdot x_i^p \geq S_t + S_t^{th}, \forall t \in T \tag{3}$$

TABLE I: Descriptions of Parameters, Sets & Variables used in the problem formulation.

Parameter Symbol	Parameter Description
f^p	Number of agile services from the P_{th} CSP
c_i^p	cost of renting i_{th} instance at P_{th} CSP
w^p	agility factor of P_{th} CSP
u^p	number of CPUs assigned with P_{th} CSP by optimizer in a template composition
m^p	membership distribution factor for p^{th} platform
F	Super set of all agile services identified by the user
A_i^p	max number of instances of type i available at CSP p
R_{it}^p	resource t_{th} available with i_{th} instance of P_{th} resource
S_t	specification requirement of type t
S_t^{th}	threshold on resource of type t
Δ_{p1p2}	binary number indicating compatibility between CSPs
M	large integer number
Set Symbol	**Set Description**
I^P	total number of instances in P_{th} provider
P	total number of provider
T	total type of resources
Variable Symbol	**Variable Description**
$x_i^p \in [0, A_i^p]$	is an integer variable denoting the number of instances of type i at provider p
$\delta_{P_1 P_2} \in \{0, 1\}$	is a binary variable ensuring only one of incompatible P1, P2 is selected

$$\begin{aligned} &\sum_{i=1}^{I^{p_1}} x_i^{p_1} \leq M \cdot \delta_{p_1 p_2} \\ &\qquad\qquad , \forall p_1, p_2 \in P : \Delta_{p_1 p_2} = 0 \\ &\sum_{i=1}^{I^{p_2}} x_i^{p_2} \leq M \cdot (1 - \delta_{p_1 p_2}) \end{aligned} \tag{4}$$

4) Security

We consider security policy incompatibilities between CSPs as a constraint to multi-cloud resource selection for users. Security requirements are unique to groups of users and can range from authentication, access control, sensitive data storage, administrative privileges, location of CSP and communication link between nodes, and more. To mitigate such security policy issues, OnTimeFLC allow users to be able to custom-define an interoperability matrix as per their expertise and preferences. To evaluate agility for CSPs by the CSUs, we have considered four sub-factors namely: *(i) Compliance, (ii) Access Controls, (iii) Auditability, and (iv) Encryption* as shown in Figure 3.

B. Fuzzy Engineering

To guide non-expert users with resource allocation for application workflows, we deploy a fuzzy engineering model and utilize users expertise along with knowledge base of benchmarks rules for assessing PACS criteria of CSPs. Fuzzy logic is used to model uncertainty in unquantifiable variables of a system.

1) Fuzzification:

This is a process of converting "crisp values" (i.e., a user specified input or approval on a pre-defined scale) into linguistic fuzzy values. The crisp data provided by the CSUs is mapped to a fuzzy set which contains the membership functions and linguistic values corresponding to the input as shown in Table II. A fuzzy set is defined by the members it contains as shown in x ϵ X where x is the element. In particular, a fuzzy set is defined by ordered pairs as shown below:

$$A = \{(x, A(x)) \mid x \epsilon U\} \tag{5}$$

where U is the universe of discourse which contains all of the elements that may be used in fuzzy set A. The membership functions are read as A(x) wherein a membership value ranges between interval [0,1] to each element in U. In our proposed approach, the linguistic values represent the fuzzy sets $fs\ (i = b, p, a, g, e)$ which consists of bad, poor, average, great, and exceptional. Each of these variables can be interpreted differently by CSUs. Therefore, to standardize the fuzzification the membership functions are used within in the fuzzy sets to define the variation in interpretation. In our approach, CSUs can utilize linguistic values as shown in Table II in order to make a decision as to which CSP would better fit their needs.

There are multiple factors that affect individual criteria of PACS evaluation of a CSP. We identify four sub-factors for each of the PACS criteria used in our multi-stage fuzzy-logic approach as shown in Figure 3.

TABLE II: Fuzzy model linguistic terms descriptions.

Linguistic Terms	Membership Degree	Membership Description
Exceptional (e)	80-100	Near Flawless (Generally Cheap)
Great (g)	60-80	Better than most (Often Cheap)
Average (a)	40-60	Not Sure(Infrequently Cheap)
Poor (p)	20-40	Sometime fails (Frequently Costly)
Bad (b)	0-20	Frequent failures (Generally Costly)

2) Membership Function:

Membership function grades the association of a value to a set. Different criteria or variables can follow different type of membership depending on their actual distribution of effectiveness in a metric scale.

$$\mu_{A^i}(x) = e^{-(x-\mu)^2/2\sigma^2} \tag{6}$$

where μ and σ are center and width of the i^{th} fuzzy set A^i

For our problem domain, we assumed that each of PACS criteria and sub-factors will have gaussian membership function. We also control the gaussian curves parameters depending on CSUs feedback to create a custom membership function.

TABLE III: Centers and Sigma values of membership functions for sub-factors of PACS.

Criteria	μ	σ
Exceptional(e)	83.3	5
Great(g)	66.64	5
Average(a)	49.98	5
Poor(p)	33.32	5
Bad(b)	16.66	5

3) Inference Engines:

The inference engine is the core of fuzzy logic where the inference rules are applied to the fuzzy input in order to generate the fuzzy output. Essentially, the inference rules are used to evaluate the linguistic values generated from crisp fuzzy input and map them to a fuzzy set. These fuzzy sets

are then transformed into resulting output crisp values using a defuzzification process. Inference engines are superficially classified into two types: *(i) Mamdani inference system:* This inference system is intuitive and well-suited to human input and is based on an interpretable rule base. Due to its application in simulating human like thoughts based on constraints, we have utilized it to learn about user biases towards different CSPs. Each of these inference rules is composed of if-then statements wrapped around linguistic terms (Table IV). The if-then rules contain the antecedents (i.e., linguistic input terms) and the consequence(i.e., linguistic output). When fabricating an inference rule, operators such as "and," "or," and sometimes "not" are used [18]. For our proposed model, we have used primarily "and" operator which is defined as below.

$$\mu A \cap B(x) = min[\mu A(x), \mu B(x)] \tag{7}$$

This rule extracts the minimum number of the membership values of the fuzzy sets to compute the "and" operation

(ii) Sugeno inference system: This inference uses singleton output membership functions that are either constant or a linear function of the input values. The defuzzification process for a Sugeno system is more computationally efficient compared to that of a Mamdani system, since it uses a weighted average or weighted sum of a few data points rather than compute a centroid or center of gravity of a two-dimensional area. We used the Mamdani inference system since we primarily aim to assess CSU's cloud platform preferences by translating and validating their intuition and human reasoning with the knowledge of experts as discussed in [5][29].

4) Defuzzification

During defuzzification, the fuzzy output from the inference engine is mapped to a crisp value that provides the most accurate representation of the fuzzy set [5]. The fuzzy outputs are represented as F_{o1} $(o1 = b, p, a, g, e)$ for the first level and F_{o2} $(o2 = AWS, GENI, MU)$ for the second level and, they share the same gaussian membership function. The system involves five types of defuzzification methods for interpretations of rules namely: a) centroid, b) center of gravity, c) bisector of the area, d) largest of maximum, and e) smallest of maximum. We use the centroid method to obtain the crisp outputs from our fuzzy inference engine [30] which is represented in below equation -

$$Crisp\ Fuzzy\ Output = \frac{\sum_{p=1}^{P} z_p \cdot u_c(z_p)}{\sum_{p=1}^{P} u_c(z_p)} \tag{8}$$

The centroid method as shown in Equation 8 and Figure 5 uses the center of mass, which is represented as z, in a fuzzy output distribution to determine a single scalar value. The membership of the fuzzy sets is presented in u_c, whereas the value of the membership is represented as z_p. Finally, the crisp output from the defuzzifier is an approximation that is used to represent the PACS-index of a CSP based on the evaluation of the first-level factors by the CSU. The CSUs can then use theses indexes of a CSP to review if the PACS of the CSP is sufficient enough for their needs through the second level of fuzzy inference.

5) Creating Rule knowledge Base:

Fuzzy logic inference works in synchronization with rules given by users as well as base set of rules. To collect base set of rules, we created an online real time data collection approach following below methodologies:

- User identifies themselves as expert or non-expert workflow users.
- Rule data is collected only for expert users.
- Rules are created to assess different available CSPs on a predefined scale e.g., 1-100, for non-functional PACS criteria as well as sub-factors affecting them.

Two sets of rules used in our evaluation of cloud PACS in linguistic terms are listed in Table V. One of these rule set is applied at the fist level of inference, while the second set is applied at the second level of inference. The full abbreviation of each variable used in established rules is presented in Table IV.

TABLE IV: Abbreviations used in fuzzy rules.

Parameter	Abbreviation	Parameter	Abbreviation
Performance	P	Auto-Scaling	AS
Agility	A	Capacity	CT
Cost	C	Compute	CC
Security	S	GPU	GC
Availability	AV	Storage	SC
Reliability	RL	RAM	RC
Response Time	RT	Compliance	CE
Throughput	TP	Access Controls	AC
Automation	AN	Auditability	AY
Flexibility	FY	Encryption	EN

TABLE V: Rules for evaluating the CSPs' PACS using a Linguistic Form.

Rule No.	Rules description for CSP inference
1	IF (P $\approx$ e) $\rightarrow$ CSP $\cong$ AWS
2	IF (P $\approx$ g) $\rightarrow$ CSP $\cong$ GENI
3	IF (P $\approx$ p) $\rightarrow$ CSP $\cong$ MU
4	IF (A $\approx$ e) $\rightarrow$ CSP $\cong$ AWS
5	IF (A $\approx$ g) $\rightarrow$ CSP $\cong$ GENI
6	IF (A $\approx$ p) $\rightarrow$ CSP $\cong$ MU
7	IF (C $\approx$ p) $\rightarrow$ CSP $\cong$ AWS
8	IF (C $\approx$ a) $\rightarrow$ CSP $\cong$ GENI
9	IF (C $\approx$ e) $\rightarrow$ CSP $\cong$ MU
10	IF (S $\approx$ e) $\rightarrow$ CSP $\cong$ AWS
11	IF (S $\approx$ a) $\rightarrow$ CSP $\cong$ MU
12	IF (S $\approx$ a) $\rightarrow$ CSP $\cong$ GENI
Rule No.	**Rules description for CSPs PACS inference from subfactors**
1	IF (AV $\approx$ e) $\wedge$ (TP $\approx$ g) $\rightarrow$ P $\cong$ e
2	IF (RL $\approx$ e) $\wedge$ (RT $\approx$ g) $\rightarrow$ P $\cong$ e
3	IF (RT $\approx$ p) $\wedge$ (AV $\approx$ e) $\rightarrow$ P $\cong$ g
4	IF (AN $\approx$ e) $\wedge$ (FY $\approx$ g) $\rightarrow$ A $\cong$ e
5	IF (FY $\approx$ g) $\wedge$ (AS $\approx$ g $\rightarrow$ A $\cong$ g
5	IF (AS $\approx$ p) $\wedge$ (CT $\approx$ p) $\rightarrow$ A $\cong$ p
6	IF (CC $\approx$ p) $\wedge$ (GC $\approx$ a) $\rightarrow$ C $\cong$ e
7	IF (GC $\approx$ a) $\wedge$ (RC $\approx$ a) $\rightarrow$ C $\cong$ a
8	IF (SC $\approx$ g) $\wedge$ (CC $\approx$ b) $\rightarrow$ C $\cong$ p
9	IF (CE $\approx$ e) $\wedge$ (AY $\approx$ a) $\rightarrow$ S $\cong$ g
10	IF (AC $\approx$ a) $\wedge$ (EN $\approx$ a) $\rightarrow$ S $\cong$ a
11	IF (AY $\approx$ b) $\wedge$ (EN $\approx$ p) $\rightarrow$ S $\cong$ b

6) Validating Gaussian Membership with CSUs Rule Base:

Through validation from multiple iteration from users, we consider that the membership function for CSPs follow gaussian membership function. For simplicity, we assume that the sub-factors contributing to PACS criteria of any CSP follows gaussian membership which is justified as most of the human

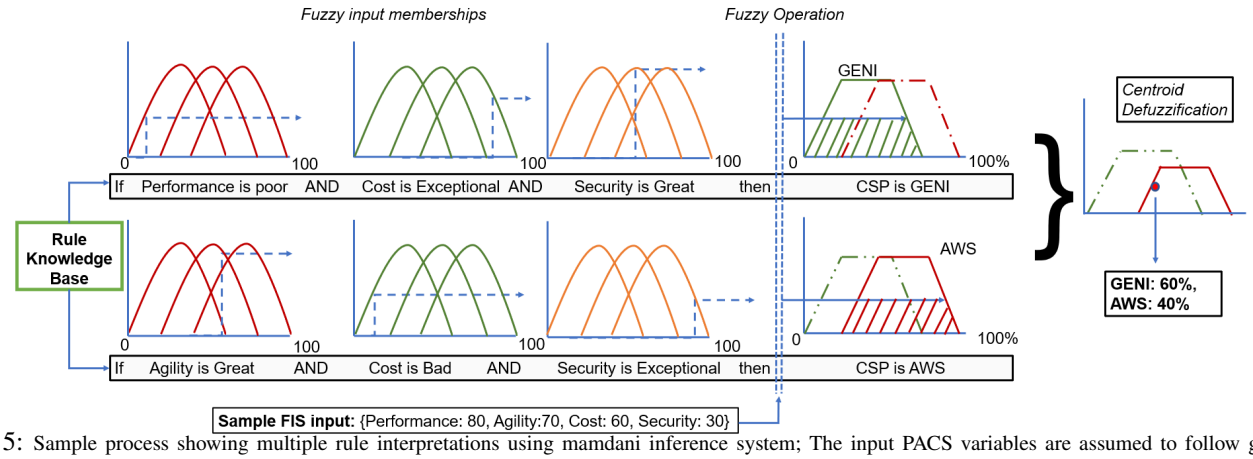

Fig. 5: Sample process showing multiple rule interpretations using mamdani inference system; The input PACS variables are assumed to follow gaussian membership function and the CSPs membership function is shown on a trapezoidal function.

intuitions when validated for large number of people follows gaussian distribution [31]. We then aggregate the fuzzy outputs distribution from 1000 CSUs from PACS subfactors to extract the performance, agility, cost and security index score that we term as "PACS-index" for each CSP. This simulation process for 1000 CSUs generates PACS-index values for each of the candidate CSPs and approximately fit into gaussian models. Thus, we get gaussian membership functions for each of the PACS criteria with different c_i and σ_i distribute on a scale from 1-100. We repeat the simulation again in second stage of the Fuzzy modelling where the output is a CSP. The membership function for the input variable in second stage i.e., PACS is aggregated from the previous step.

7) Membership distribution

Once the fuzzy engineering model shown in Figure 2 is trained, the model returns a membership distribution for selection of CSPs for a specific expert user inputs. To bias the optimizer's objective function towards a CSP using membership distribution, we formulate below formula -

$$Membership\ Factor\ (m^p) = (1 - M^p) \tag{9}$$

where, M^p is the membership value of p^{th} platform obtained from the fuzzy engineering model. This formulation ensures that the effective cost of instances from a platform with higher membership distribution is reduced in the objective function represented in Equation 2. Since the maximum value of membership distribution for any CSP is 1, hence CSPs with lower membership value will have lower reduction in effective price, and thus this formulation holds validity for different distributions.

V. Performance Evaluation

A. Experiment Testbed Setup

We evaluate the OnTimeFLC framework architecture as shown in Figure 4, where we integrated FIS into the KB-Commons web portal to collect user specifications and fuzzy input. We also implemented the optimizer engine middleware that was hosted on an independent GENI [7] node machine as a web service. Pegasus [22], HTCondor [23], pyGlidein [24] and Cyverse iCommand were installed and configured on the independent GENI node. Application workflows as shown in Table VI were created using Pegasus and configured in the node machine as illustrated in Figure 4. Cyverse iCommand and Pegasus APIs were used to fetch and stage the user data from Cyverse portal to CSPs for processing the workflows.

B. Fuzzy Model Accuracy Evaluation

Herein, we evaluate accuracy of the multi-level fuzzy engineering model and its ability in evaluating cloud platforms from users perspective by validations. For the evaluation of the fuzzy model, we assume that the user is an expert user. We create the fuzzy model system and create rules defining behaviors of our three base cloud service providers i.e., MU (private cloud), GENI (community cloud), AWS (public cloud).

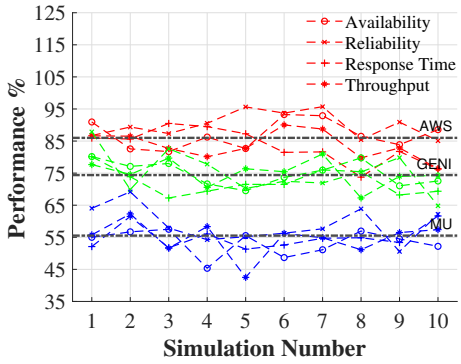

Fig. 6: Average performance measurement for CSPs using the fuzzy model with simulation on 1000 CSUs for 10 iterations.

The PACS-index of CSPs can be calculated based on the level of satisfaction a CSU receives from a given CSP. More specifically, we ensure the validity for each of these PACS-index for CSPs by applying the theorem as given by [18]. As per the theorem suggested in [18], we sample inputs from only those CSUs who are experts i.e., their crisp input for cloud platform services align with true service level agreements from cloud platforms. Such a process of identifying expert CSUs needs external independent third party validation. For our solution, we verify expert users by cross-checking there

TABLE VI: Application workflows used in evaluation experiments.

Workflow Name	Workflow Description	Resources Data(Gb), vCPU, RAM(Gb), Network(Gbps), Clock
FactQC	FastQC Quality Check workflow is used conduct the quality control checks on raw sequencing data so that we can remove some low score data before the next step analysis.	{3, 4, 5, 0.5, 1}; {4, 8, 10, 1, 1.2}
RNA-Seq	RNA-Seq analysis allow us to identify the differential expressed genes by performing the pair-wise comparison of experimental groups/ conditions of sequencing data.	{10, 12, 20, 2, 1.2}; {10, 16, 25, 4, 1.4}
PGen	PGen workflow [2] allow users to identify the single nucleotide polymorphisms (SNPs: a substitution of a single nucleotide that occurs at a specific position in the genome) and insertion-deletion (indels: an insertion or deletion of nucleotides from a sequence) and perform SNP annotation.	{20, 20, 50, 8, 1.4}; {20, 24, 100, 12, 2}; {30, 28, 120, 16, 2.2}

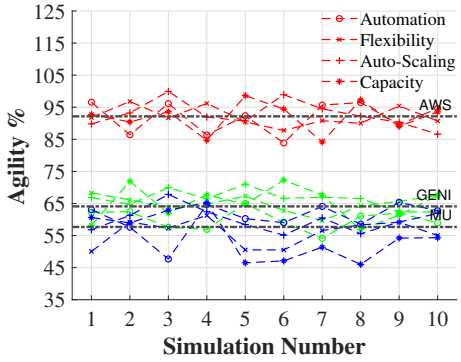

Fig. 7: Average agility measurement for CSPs using the fuzzy model with simulation on 1000 CSUs for 10 iterations.

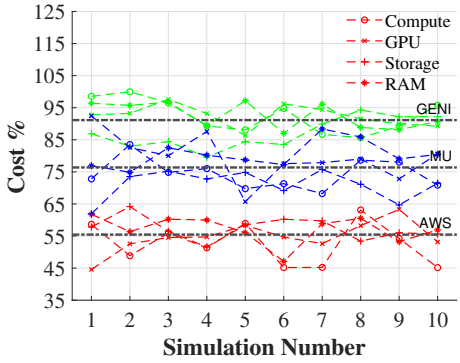

Fig. 8: Average cost measurement for CSPs using the fuzzy model with simulation on 1000 CSUs for 10 iterations.

choices to align with a large number of CSUs. To validate the effectivity of our proposed solution, we simulate 1000 CSUs iteratively 10 times with inherent biases for subfactors as shown in Table IV towards three CSPs namely AWS, GENI and MU.

Figures 6, 7 and 8 shows our simulation towards Performance, Agility and Cost benchmarking of three concerned CSPs. Each data point in Figures 6, 7 and 8 represents the average approval in % towards specific sub-factor of the CSPs. We simulate the process 10 times, and we find the fuzzy output distribution toward PACS of CSPs from the fuzzy inputs in each iteration. For example, considering only performance, each CSU in each iteration results in one fuzzy output so we get 1000 performance fuzzy values using centroid defuzzification (note that the membership functions are gaussian) for each CSP. We remark that we have created the fuzzy models to simulate defuzzification methods using the Matlab Fuzzy Toolbox [32]. When this process is iterated 10 times, we get 10000 performance fuzzy values for each CSP. The average of these fuzzy values for these three CSPs is shown in the Figures 6, 7 and 8 as straight horizontal lines. Note that these fuzzy values have a range of 1 to 100. When this scale is divided into 5 subdivisions as mentioned in Table III, it provides intuition for rule creation in the second level of fuzzy inference as shown in Table V. Based on these results, we prove the validity of our proposed fuzzy inference system which can guide a new user towards CSPs based on their preferences towards sub-factors.

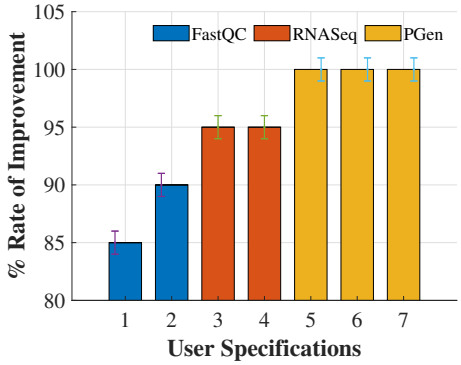

Fig. 9: Percentage of workflows with improved execution time (i.e., lesser time) when fuzzy engineering was used 'with' OnTimeFLC resource broker vs. OnTimeFLC was used 'without' fuzzy inference inputs.

C. *OnTimeFLC evaluation:*

We now describe the evaluation results of our proposed framework OnTimeFLC's efficiency in composing cost-effective templates solutions for any given user requirements and preferences. We evaluate the efficiency of fuzzy model to improve application workflows execution time performance at different resource specifications as shown in Table VI. We compose, allocate and compare template solution performance for the user specifications for two cases:(i) Fuzzy model with expert user was considered to improve execution performance of workflows by selection of CSPs using OnTimeFLC and, (ii) Only OnTimeFLC's core ILP optimizer engine. The experiments were repeated (10 times) iteratively for each specification to calculate average execution time for workflows. Figure 9 corresponds to our experimental results that show the time to execute the workflow reduced in 98% of the cases with OnTimeFLC including fuzzy engineering. We remark that the rate of improvement in efficiency is highly dependent on expertise of user as the fuzzy model suggests CSPs considering users PACS rating of CSPs.

VI. Conclusion

In this paper, we proposed a novel middleware (i.e., OnTimeFLC) based on fuzzy engineering to utilise expertise of users for better cloud resource selection. The method is composed of a multi-level fuzzy model based on factors of performance, agility, cost and security (PACS) which aids a resource broker based on integer linear programming in composing multi-cloud solution templates. We validated that the proposed fuzzy engineering approach by simulating decision making and expertise utilization of users in improving selection of appropriate CSPs. We also showed how OnTimeFLC can help with the resource management in a science gateway deployment viz., KBCommons to help bioinformatics researchers/educators. Our resource brokering experiment featured a dataset of more than 300 instance from multiple CSPs i.e., corresponding to three CSP resources including AWS (public cloud), GENI (community cloud) and MU (private cloud).

Future work can include live system deployment to capture real user preferences towards CSPs and validations through an independent auditor. Further, machine learning can be used to improve rule creations for our fuzzy inference system.

Acknowledgment

This work was supported by the National Science Foundation under award number OAC-1827177. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] Pandey A., Lyu Z.,Joshi T., and Calyam P., "OnTimeURB: Multi-Cloud Resource Brokering for Bioinformatics Workflows," 2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), San Diego, CA, USA, 2019, pp. 466-473.

[2] Liu, Yang & Khan, Saad & Wang, Juexin & Rynge, Mats & Zhang, Yuanxun & Zeng, Shuai & Chen, Shiyuan & Maldonado dos Santos, Joao Vitor & Valliyodan, Babu & Calyam, Prasad & Merchant, Nirav & Nguyen, Henry & Xu, Dong & Joshi, Trupti. (2016). "PGen: Large-scale genomic variations analysis workflow and browser in SoyKB" *BMC Bioinformatics*, 17. 337. 10.1186/s12859-016-1227-y.

[3] Amazon OpsWorks. Available: https://aws.amazon.com/opsworks/ [Online][Last accessed: 14th December 2020]

[4] Akhtar, Samia. (2014). Performance Evaluation In Cloud Computing Using Fuzzy Logic.

[5] Ebrahimnejad, Ali & Verdegay, José Luis, "Fuzzy Set Theory," Fuzzy Sets-Based Methods and Techniques for Modern Analytics, vol. 364

[6] Amazon Web Services. Available: https://aws.amazon.com/ [Online][Last accessed: 14th December 2020]

[7] Berman, M., Chase, J.S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R. and Seskar, I., 2014. GENI: A federated testbed for innovative network experiments. Computer Networks, 61, pp.5-23.

[8] MU Data Center. Available: https://doit.missouri.edu/services/servers-administration/server-housing/mu-data-center/ [Online][Last accessed: 14th December 2020]

[9] Antequera, Ronny & Calyam, Prasad & Ankathatti Chandrashekara, Arjun & Mitra, Reshmi, "Recommending heterogeneous resources for science gateway applications based on custom templates composition", *Future Generation Computer Systems*, 100. DOI: 10.1016/j.future.2019.04.049, 2019

[10] Bahuti, Marcelo & Abreu, Lucas & Yanagi Junior, Tadayuki & de Lima, Renato & Campos, Alessandro. (2018). Performance of fuzzy inference systems to predict the surface temperature of broiler chickens. Engenharia Agrícola. 38. 813-823. 10.1590/1809-4430-eng.agric.v38n6p813-823/2018.

[11] Hayat, Bashir & Kim, Kyong & Kim, Ki-Il. (2018). A study on fuzzy logic based cloud computing. Cluster Computing. 21. 1-15. 10.1007/s10586-017-0953-x.

[12] Chopra, Pooja & Bedi, RPS. (2014). Fuzzy Logic Based Resource Provisioning in Cloud Computing. Advanced Engineering Technology and Application. Adv. Eng. Tec. Appl. 8, No. 1, 1-9 (2019). http://dx.doi.org/10.18576/aeta/080101

[13] Bhardwaj, Tushar & Sharma, Subhash. (2018). Fuzzy logic-based elasticity controller for autonomic resource provisioning in parallel scientific applications: A cloud computing perspective. Computers & Electrical Engineering. 70. 1049-1073. 10.1016/j.compeleceng.2018.02.050.

[14] Mehranzadeh, Amin & Hashemi, Seyyed Mohsen. (2013). A Novel-Scheduling Algorithm for Cloud Computing based on Fuzzy Logic. International Journal of Applied Information Systems. 5. 28-31. 10.5120/ijais13-450939.

[15] Zavvar, Mohammad & Rezaei, Meysam & Garavand, Shole & Ramezani, Farhad. (2016). Fuzzy Logic-Based Algorithm Resource Scheduling For Improving The Reliability Of Cloud Computing. Asia-Pacific Journal of Information Technology & Multimedia. 05. 39-48. 10.17576/apjitm-2016-0501-04.

[16] Braiki, Khaoula & Youssef, Habib. (2019). Fuzzy-logic-based multi-objective best-fit-decreasing virtual machine reallocation. The Journal of Supercomputing. 10.1007/s11227-019-03029-8.

[17] Toosi, Adel. (2015). A Fuzzy Logic-based Controller for Cost and Energy Efficient Load Balancing in Geo-Distributed Data Centers. 10.1109/UCC.2015.35.

[18] Rizvi, Syed. & Mitchell, John & Razaque, Abdul & R. Rizvi, Mohammad and Williams, Iyonna . "A fuzzy inference system (FIS) to evaluate the security readiness of cloud service providers." Journal of Cloud Computing 9 (2020): 1-17.

[19] Knowledge Base Commons. Available: http://kbcommons.org/ [Online][Last accessed: 26th August 2020]

[20] Zeng, Shuai & Lyu, Zhen & Narisetti, Siva & Xu, Dong & Joshi, Trupti. (2019). Knowledge Base Commons (KBCommons) v1.1: A universal framework for multi-omics data integration and biological discoveries. BMC Genomics. 20. 10.1186/s12864-019-6287-8.

[21] Zeng, Shuai & Lyu, Zhen & Narisetti, Siva & Xu, Dong & Joshi, Trupti. (2018). Knowledge Base Commons (KBCommons) v1.0: A multi OMICS' web-based data integration framework for biological discoveries. 589-594. 10.1109/BIBM.2018.8621369.

[22] Deelman, Ewa & Vahi, Karan & Juve, Gideon & Rynge, Mats & Callaghan, Scott & Maechling, Philip & Mayani, Rajiv & Chen, Weiwei & Ferreira da Silva, Rafael & Livny, Miron & Wenger, Kent. (2014). Pegasus, a workflow management system for science automation. Future Generation Computer Systems. 46. 10.1016/j.future.2014.10.008.

[23] HTCondor Available: https://research.cs.wisc.edu/htcondor/index.html [Online][Last accessed: 14th December 2020]

[24] WIPACrepo-pyglidein [Online][Last accessed: 26th August 2020] Available: https://github.com/WIPACrepo/pyglidein#pyglidein DOI: 10.5281/zenodo.1469022

[25] CyVerse — Cyberinfrastructure for Data Management and Analysis. Available: https://www.cyverse.org/ [Online][Last accessed: 14th December 2020]

[26] Merchant, Nirav, et al., "The iPlant Collaborative: Cyberinfrastructure for Enabling Data to Discovery for the Life Sciences," *PLOS Biology* (2016), doi: 10.1371/journal.pbio.1002342.

[27] IBM ILOG CPLEX Optimization Studio. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio [Online][Last accessed: 14th December 2020]

[28] Cloud Computing Comparison Engine Available: https://www.cloudorado.com/ [Online][Last accessed: 14th December 2020]

[29] Topaloglu, Fatih & Pehlivan, Hüseyin. (2018). Comparison of Mamdani type and Sugeno type fuzzy inference systems in wind power plant installations. 1-4. 10.1109/ISDFS.2018.8355384.

[30] Chakraverty, Snehashish & Sahoo, Deepti Moyi & Mahato, Nisha Rani (2019) Defuzzification. Springer:117– 127

[31] Shen, Yelong & Jin, Ruoming & Dou, Dejing & Chowdhury, Nafisa & Sun, Junfeng & Piniewski, Brigitta & Kil, David. (2012). Socialized Gaussian Process Model for Human Behavior Prediction in a Health Social Network. Proceedings - IEEE International Conference on Data Mining, ICDM. 1110-1115. 10.1109/ICDM.2012.94.

[32] MathWorks 2017a https://in.mathworks.com/company/newsroom/mathworks-announces-release-2017a-of-the-matlab-and-simulink-pro.html [Online][Last accessed: 14th December 2020]