Agri-Info: Cloud Based Autonomic System for Delivering Agriculture as a Service

Sukhpal Singh[1], Inderveer Chana[2] and Rajkumar Buyya[3]

[1]School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
[2]Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, Punjab, India
[3]Cloud Computing and Distributed Systems (CLOUDS) Laboratory,
School of Computing and Information Systems, The University of Melbourne, Australia
s.s.gill@qmul.ac.uk, inderveer@thapar.edu, rbuyya@unimelb.edu.au

ABSTRACT

The Internet of Things (IoT) and cloud computing paradigms offer enhanced services for agricultural applications to manage the data efficiently. To provide an effective and reliable agriculture as a service, there is a need to manage Quality of Service (QoS) parameters to efficiently monitor and measure the delivered services dynamically. This paper presents a QoS-aware cloud based autonomic information system called *Agri-Info* for delivering agriculture related information as a service through the use of latest Internet-based technologies such as cloud computing and IoT which manage various types of agriculture related data based on different domains of agricultural industry. Proposed system gathers information from various users through preconfigured IoT devices (mobiles, laptops or iPads). It further manages and delivers the required information to users and diagnoses the agriculture status automatically. We have developed the web and mobile-based application and evaluated the performance of the proposed system in cloud environment using CloudSim toolkit based small scale environment. Results demonstrate our system yields in a reduction on 12.46% cost, on 15.52% network bandwidth, on 10.18% execution time and 13.32% in latency. Furthermore, a case study of an Indian village is presented to identify the customer satisfaction of farmers.

Keywords - Cloud computing, Autonomic, Quality of Service, Agriculture as a Service, Resource Scheduling, Case Study, Cloud Application

1. Introduction

Cloud computing paradigm offers an effective application platform, which manages and delivers the new emerging Internet of Things (IoT) based applications in the field of healthcare, agriculture, education or finance efficiently over the Internet. Though, delivering dedicated cloud services that ensure application's dynamic Quality of Service (QoS) requirements and user satisfaction is a big research challenge in cloud computing [1]. As dynamism, heterogeneity and complexity of applications is increasing rapidly, this makes cloud systems unmanageable in-service delivery. To overcome these problems, cloud systems require self-management of services. Autonomic cloud computing systems provide the environment in which IoT applications can be managed efficiently by fulfilling QoS requirements of applications without human involvement [2].

Emergence of IoT and ICT (Information and Communication Technologies) plays an important role in agriculture sector by providing services through computer-based agriculture systems [3] [20]. But these agriculture systems are not able to fulfill the needs of today's generation due to lack of important requirements like processing speed, lesser data storage space, network utilization and latency [4] [21] and even the resources used in computer-based agriculture systems are not utilized efficiently [5] [22]. To solve the problem of existing agriculture systems, there is a need to develop a cloud based service [12] that can easily manage different types of agriculture related data based on different domains (crop, weather, soil, pest, fertilizer, productivity, irrigation, cattle and equipment) through these steps: i) gather data from various users through preconfigured IoT devices, ii) analyze the gathered data in an efficient manner, iii) store the classified information in cloud repository for future use, and iv) automatic diagnose of the agriculture status using concepts such as fuzzy logic [2]. In addition, cloud based autonomic information system is also able to identify the QoS requirements of user request and resources are allocated efficiently to execute the user request based on these requirements. Cloud based services can significantly improve latency, network bandwidth, execution time and customer satisfaction.

1.1 Motivation

The motivation of this paper is to design an architecture of cloud based autonomic information system for agriculture service called ***Agri-Info*** which manages various types of agriculture related data based on different domains and maintains required level of QoS. The main objectives of this research work are: i) to propose an autonomic resource management technique which is used: a) to gather the information from various users through preconfigured IoT devices, b) to analyze the information and c) to store the information in cloud repository (fuzzy rule base) for future use and e) diagnose the agriculture status automatically and ii) to allocate resources automatically at infrastructure level after identification of QoS requirements of user request. Agri-Info improves user satisfaction by fulfilling their expectations and increases availability of services.

1.2 Article Structure

The rest of the paper is organized as follows. Section 2 presents related work and contributions. Proposed architecture is presented in Section 3. Section 4 presents the graphical user interface with the help of web and mobile based application. Section 5 describes the experimental setup and presents the results of performance evaluation and a case study of Agri-Info. Section 6 presents summarizes the paper and highlights open challenges and future directions.

2. Related Work

Literature reported that existing agriculture systems [23] [24] have been developed with limited functionality. This section presents the related work of existing agriculture systems. Jirapond et al. [1] proposed control system (Smart-Farm) using node sensors in the crop field with data management via smartphone and a web application to control the watering process to the crops and helps to reduce the cost of watering crop. Alexandros et al. [6] proposed architecture of a farm management system using characteristics of Internet which focuses on procedure of farming and mechanisms to exchange the information among stakeholders. Further, this architecture describes the method for better management of only some of the tasks of farmers without using the autonomic concept. Ranya et al. [7] presented Agriculture Land Suitability Evaluator (ALSE) to study various types of land to find the appropriate land for different types of crops by analyzing geo-environmental factors. ALSE used Global Information System (GIS) capabilities to evaluate land using local environment conditions through digital map and based on this information decisions can be made. Raimo et al. [25] proposed Farm Management Information System (FMIS) used to find the precision agriculture requirements for information systems through web-based approach. Author identified the management of GIS data is a key requirement of precision agriculture. Sorensen et al. [8] studied the FMIS to analyze dynamic needs of farmers to improve decision processes and their corresponding functionalities. Further, they reported that identification of process used for initial analysis of user needs is mandatory for actual design of FMIS.

Sorensen et al. [9] identified various functional requirements of FMIS and information model is presented based on these requirements to refine decision processes. They identified that complexity of FMIS is increasing with increase in functional requirements and found that there is a need of autonomic system to reduce complexity. Shitala et al. [10] presented mobile computing-based framework for agriculturists called AgroMobile for cultivation and marketing and analysis of crop images. Further, AgroMobile is used to detect the disease through image processing and also discussed how dynamic needs of user affects the performance of system. Seokkyun et al. [11] proposed cloud-based Disease Forecasting and Livestock Monitoring System (DFLMS) in which sensor networks has been used to gather information and manages virtually. DFLMS provides an effective interface for user but due to temporary storage mechanism used, it is unable to store and retrieve data in databases for future use.

2.1 Comparisons of Agri-Info with Existing Agriculture Systems

The proposed QoS-aware cloud based autonomic information system (Agri-Info) has been compared with existing agriculture systems as described in Table 1, which clearly shows the novelty and superiority of the Agri-Info.

Table 1: Comparisons of existing agriculture systems with proposed system (Agri-Info)

Agriculture System	Domains	Autonomic	QoS-aware	Fuzzy Logic	Cloud Resource Management	Graphical User Interface	Case Study
Smart-Farm [1]	Irrigation	X	X	X	X	✓	X
ALSE [7]	Soil	X	X	X	X	X	X
FMIS [25]	Pest and Crop	X	X	X	X	X	X
AgroMobile [10]	Crop	X	X	X	X	X	X
DFLMS [11]	Crop	X	X	X	X	X	X
Agri-Info (Our Work)	Crop, Weather, Soil, Pest, Fertilizer, Productivity, Irrigation, Cattle and Equipment	✓	✓	✓	✓	✓	✓

2.2 Previous Work and Our Contributions

This research work is an extended and summarized version of our previous work [12], (published in The Journal of Organizational and End User Computing) and an updated version of our Technical Report[1]. In our previous work, we proposed a cloud-based system for agriculture, which gathers and manages data coming from systems or IoT devices and provides the required information to users based on their requirement. However, these works are not able to make smart and intelligent decisions to diagnose the status of agriculture related user query. There is a need of rule based intelligent module, which can diagnose the status of user related query automatically within their deadline and minimum value of latency,

[1] CLOUDS-TR-2015-2, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, 2015, web link: https://arxiv.org/abs/1511.08986

network bandwidth and execution time. In this paper, we use Fuzzy logic to make the decisions based on defined rules to diagnose the status as well as schedule the resources automatically by optimizing the execution time, cost, latency and network bandwidth. Further, we have designed web-based and mobile-based application for user through which user or farmer can interact with the system efficiently. The major contributions of this work are:

- To propose cloud based autonomic model for offering Agriculture-as-a-Service through web and mobile based application and maintains the level of QoS.
- To gather the information from various users through preconfigured IoT/Edge devices.
- To diagnose the agriculture status automatically using Fuzzy Logic, which helps to make further optimal decisions.
- To execute cloud resources automatically and improves user satisfaction and usability of cloud service automatically.
- To evaluate the performance of Agri-Info with existing agriculture system i.e. Smart-Farm [1] based on various QoS parameters such as execution time, cost, latency and network bandwidth.
- To validate Agri-Info with the real case study of an Indian Village based on the customer satisfaction level of farmers.

3. Agri-Info Architecture

Existing agriculture systems are not able to fulfill the needs of today's generation due to missing of important requirements like processing speed, lesser data storage space or latency and even the resources used in computer-based agriculture systems are not utilized efficiently. To solve the problem of existing agriculture systems, there is a need to develop a cloud based autonomic information system which delivers Agriculture as a Service (AaaS). In this section, we present an architecture of QoS-aware cloud based autonomic information system for agriculture service called *Agri-Info*, which manages various types of agriculture related data based on different domains. Architecture of Agri-Info is shown in Figure 1.

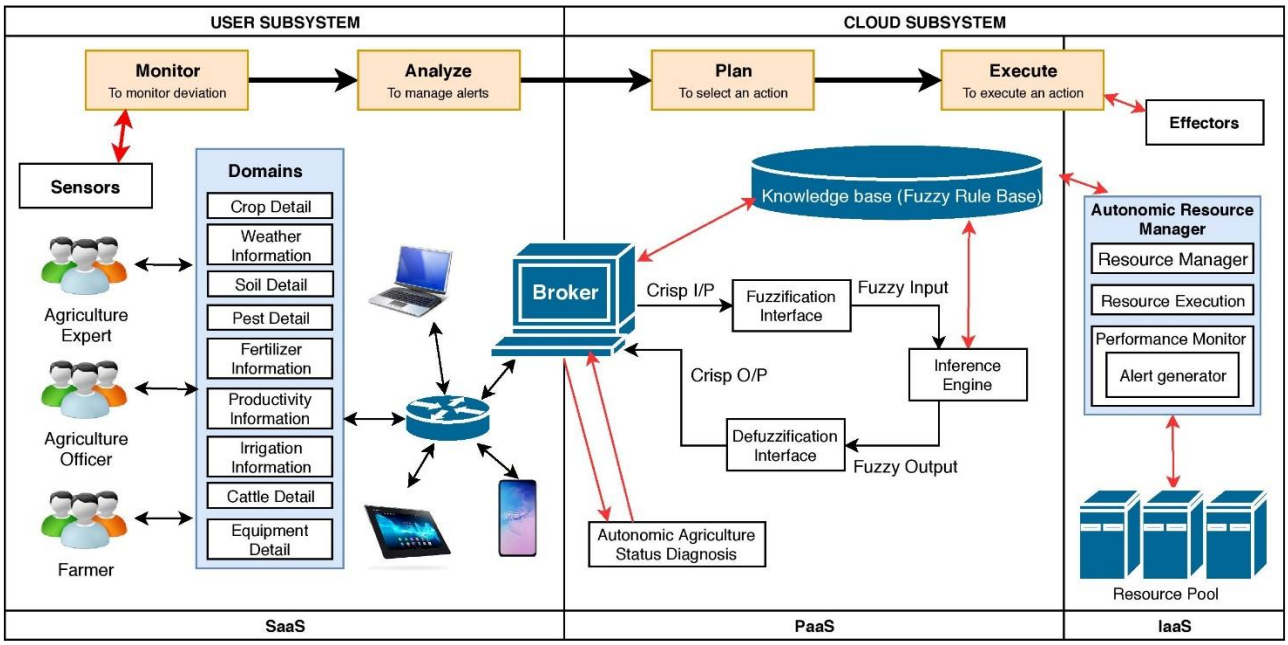

Figure 1: Agri-Info Architecture [12]

The main objectives of this proposed system are: i) to get information from various IoT devices, ii) to analyze the information using Fuzzy logic and create fuzzy rules, iii) to store the user data and fuzzy rules in cloud repository (Fuzzy Rule Base) for future decisions, iv) to respond the user queries automatically based on the information stored in Fuzzy Rule Base and v) to allocate the resources automatically based on QoS requirements of different requests. Architecture of Agri-Info comprises following two subsystems: i) user and ii) cloud.

3.1 User Subsystem

This subsystem provides user interface, in which different type of users interacting with Agri-Info to provide and get useful information about agriculture based on different domains. We have considered nine types of information of different domains in agriculture: crop, weather, soil, pest, fertilizer, productivity, irrigation, cattle and equipment. Users are basically classified in three categories: i) agriculture expert, ii) agriculture officer and iii) farmer. Agriculture expert shares professional

knowledge by answering the user queries and updates the fuzzy rule database based on the latest research done in the field of agriculture with respect to their domain. Agriculture officers are the government officials those provides the latest information about new agriculture policies, schemes and rules passed by the government. Farmer is an important entity of Agri-Info who can take maximum advantage by asking their queries and getting automatic reply after analysis. Use case diagram shown in Figure 2, describes the important functions user can perform with Agri-Info.

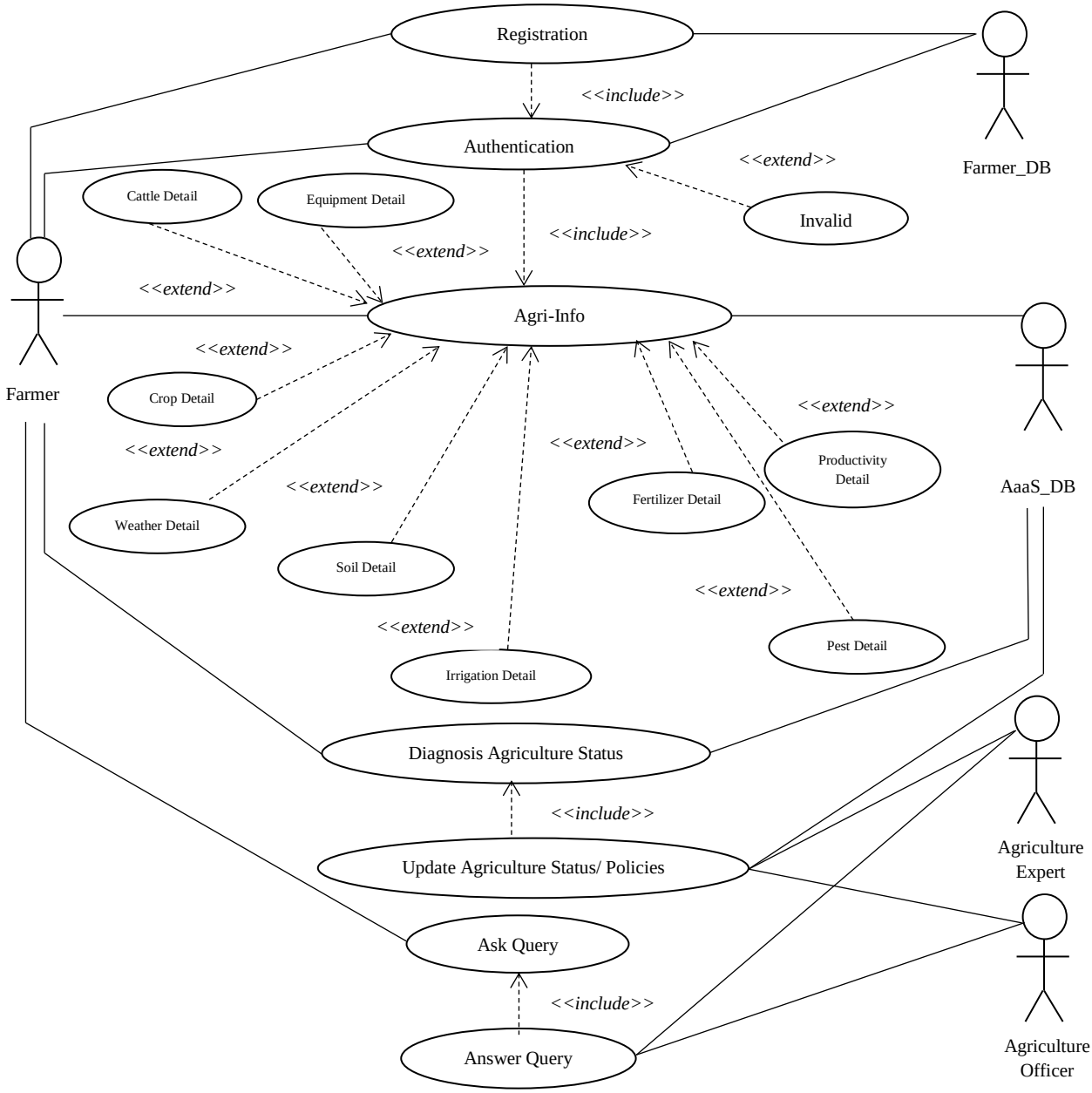

Figure 2: Use Case Diagram of Agri-Info

Use case diagram describes the interaction of users with Agri-Info. The notation <<*include*>> uses when a use case is depicted as using the functionality of another use case. For example: the functionality of use case "Agri-Info" can be accessed only with valid "Authentication". The notation <<*extend*>> uses when the next connected use case is optional. For, example: indicates that an "Invalid Password" use case may include (subject to specified in the extension) the behavior specified by base use case "Authentication". Figure 3 shows the sequence diagram of the user interaction in Agri-Info. Sequence diagram shows the: 1) collaboration of objects based on a time sequence and 2) how the objects interact with others in a particular scenario of Use Case. Firstly, successful registration of user has been demonstrated. After performing the task of the user's authentication and authorization, the home page of user will be displayed. User can write their query regarding the agriculture information required from any of the domain and then system will provide the required information after analysis of user

query automatically. All the requests are analyzed based on their information asked by user and updates the database. Users can monitor any data related to their domain and get their response without visiting the agriculture help center. It integrates the different domains of agriculture with Agri-Info. Agri-Info does not require any technical expertise to use this system. The information or queries received from user(s) are forwarded to cloud repository for updation and response send back to particular user on their preconfigured devices (tablets, mobile phones or laptops) via Internet.

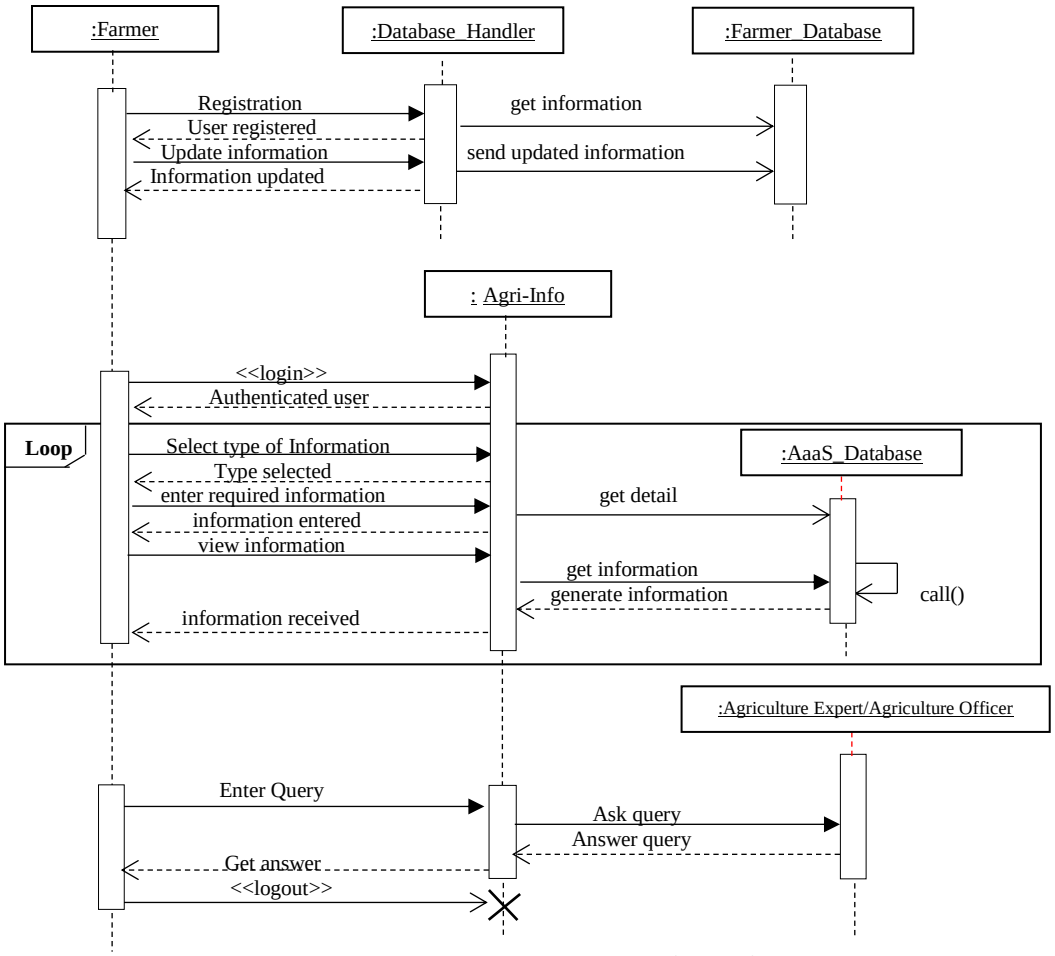

Figure 3: Sequence Diagram of Agri-Info

3.2 Cloud Subsystem

This subsystem contains the platform in which agriculture web service is hosted on a cloud [17]. Agriculture web or mobile service allows to process the agriculture information provided by users of different domains of agriculture. Agri-Info gathered different number of attributes based on specific agriculture information submitted by user through agriculture web service or mobile application as shown in Table 2.

Table 2: Domains and their Attributes [12]

Domain	Attributes
Crop Info	Cropid, Name, Type, Soil Moisture, Temperature, Season, Avg. Productivity, Min Land, Growing Period, Seed Type, Price, Quantity, Disease and Treatment.
Weather Info	Humidity, Temperature, Pressure, Wind Speed, Rainfall and Location
Soil Info	Bulk Density, Inorganic Material, Organic Material, Water, Air, Color, Texture, Structure and Infiltration
Pest Info	Type, Effect, Treatment, Solubility in Water, Outcome and Price
Fertilizer Info	Type, Nutrient Composition and Price
Productivity Info	Soil Type, Crop Type, Season, Rainfall, Pest Info, Fertilizer Info and Irrigation Info
Irrigation Info	Climate Factors (Rainfall/Temperature), Crop Type, Season and Soil Type
Cattle Info	Type, Quantity, Area, Layout and Structure of Yard, Feed, Drinking Water, Health Issue, Disease and Treatment
Equipment Info	Type, Quantity, Area, Budget, Price, Maintenance Cost and Work Type

These details are stored in cloud repository in different classes for their respective domains with unique identification number. The information is monitored, analyzed and processed continuously by Agri-Info. We have designed different classes for

every domain and sub classes for further categorization of information. In storage repository or fuzzy rule base, user data is categorized based on different predefined classes of every domain. This information is further forwarded to agriculture experts and agriculture officers for final validation through IoT devices. *Agriculture Manger* is an independent entity and in charge of this system and will be responsible to evaluate the correctness of the knowledge present in the database. Agriculture Manger can be an employee of the government or private agriculture industry. Based on QoS requirements, autonomic resource manager identifies resource requirements and allocates and executes the resources at infrastructure level. Performance monitor is used to verify the performance of system and maintain it automatically. If system will not be able to handle the request automatically then system will generate alert.

3.2.1. Fuzzy Logic Based Status Diagnose Module

Agri-Info executes the user requests to diagnose the status of their query automatically. The components of fuzzy system [13] [14] [15] used in Agri-Info are described below:

3.2.1.1 Fuzzy input and output variables

It is necessary to find the input and output parameters for fuzzy system. We have considered six attributes from Table 2 as an input: Crop Name, Temperature, Soil Texture, Season, Pesticide and Fertilizer and one output: Productivity. Based on these six attributes, Agri-Info design rules and membership functions.

3.2.1.2 Fuzzy membership functions (Inferences)

Based on these input and output variable, inference engine is making decisions. Block diagram of inference engine is shown in Figure 4. Value of membership functions can be changed based on the requirements and conditions of every user request.

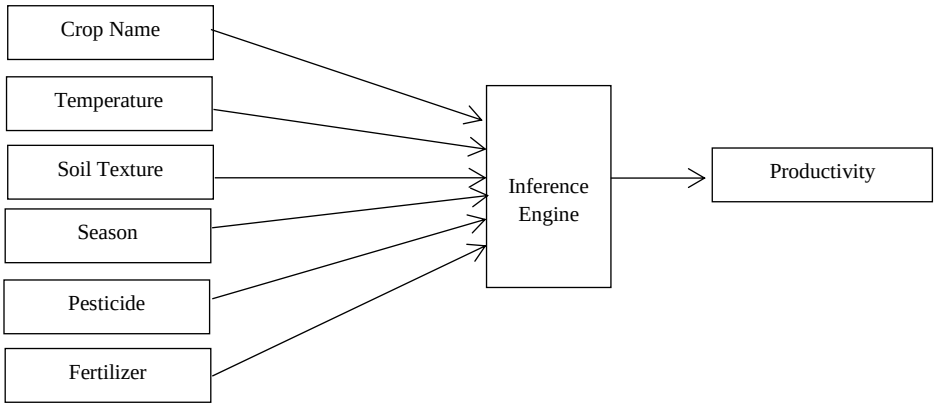

Figure 4: Block diagram of inference engine

i) Cluster and fuzzify the extracted data: To derive required member functions *(mfs)* from output values, there is a need to cluster the output values of all training instances (ti) into various clusters. After clustering, most close output values of ti belong to the similar class is considered. To achieve this objective, it further includes sub steps:

a) To find relationship among different output values of ti, Sort the output values in an ascending order

$$(ti_1 \leq ti_2 \geq \cdots \leq ti_n), \text{ where } ti_m \leq ti_{m+1} \text{ for m = 1, 2, 3…..n-1,}$$

b) To identify the similarity, there is a need to find the difference d_m between adjacent data

$$d_m = ti_{m+1} - ti_m, \text{ for each pair } ti_m \text{ and } ti_{m+1}$$

c) To transform distance d_m to a real number r_m, $r_m \in (0, 1)$

$$r_m = \begin{cases} 1 - \dfrac{d_m}{s \times sd_r}, & d_m \leq s \times sd_r \\ 0, & otherwise \end{cases}$$

Where r_m presents the similarity between ti_m and ti_{m+1} and d_m is distance between ti_m and ti_{m+1} and sd_r is standard deviation of d_m and s is a control parameter which is used to decide the shape of *mfs* of similarity.

d) Cluster the ti according to similarity

For clustering the instances, the value of μ for similarity is identified. Value of μ is used to identify the threshold value two adjacent data, where adjacent data belongs to similar class. With maximum value of μ, clusters will be smaller.

If $r_m \leq \mu$ **then** [distribute the two adjacent data into different clusters] **else** [keep two adjacent data into similar cluster]

Result is obtained in the form like: (ti_m, C_o), after the above operation. (ti_m, C_o) represents that m^{th} output data is clustered into the C_o (where C_o is o^{th} produced fuzzy region).

Suppose the value of μ is 8, then training data clustered into groups: (ti_1, C_1), (ti_2, C_1), (ti_3, C_2), (ti_4, C_1), (ti_5, C_3), (ti_6, C_1), (ti_7, C_3), (ti_8, C_2).

e) Identify the membership functions of the output space

We used membership functions for every linguistic variable. Minimum value of μ in the cluster is selected as value of membership of the two extreme points (ti_m and ti_q) of boundary to determine the membership of ti_m and ti_q. Formula used to calculate ρ_o (ti_m) and ρ_o (ti_q) is:

$$\rho_o\ (\ ti_m) = \rho_o\ (\ ti_q) = \text{minimum}\ (\ r_{m+1},\ r_{m+2}, \ldots\ldots\ldots\ r_{q-1})$$

f) Determine the value of membership which belongs to the selected cluster for every instance

Fuzzy value formed as (ti_m, C_o, ρ_{mo}) of each output data is retrieved by using above membership functions. Term (ti_m, C_o, ρ_{mo}) referred as fuzzy value (ρ_{mo}) to the cluster (C_o) of output data *(m^{th})*. Every *ti* is then represented as after transformation:

$$(\ f_1,\ f_2, \ldots\ldots\ f_h;\ (C_1,\ \rho_1), (C_2,\ \rho_2), \ldots\ldots (C_q,\ \rho_q)\ )$$

ii) Create initial membership functions for input attributes: Initial membership function is assigned to every input attribute, which is represented as: triangle (*j, k, l*) with *k - j = l - k* = the smallest predefined unit. *For example*: smallest unit is selected to be 10 if 10, 20 and 30 are three values of an attribute. We have assumed for the attribute that j_0 be its smallest value and j_n be its biggest value.

iii) Create the initial decision table: Based on initial membership functions, a multi-dimensional decision table is created in which every attribute is represented by every dimension and position's content in the decision table is treated as *Slot*. To represent the position's content of ($w_1, w_2, w_3 \ldots\ldots\ldots w_t$) in the decision table, $Slot_{(w_1,w_2,w_3\ldots\ldots\ldots w_t)}$ is created, where position value at the m^{th} dimension is represented by w_m and dimension of decision table is represented by *t.* Every slot can be empty or contain a maximum value of membership of the output data. Slots cannot be empty.

iv) Simplify the initial decision table: To eliminate redundant and unnecessary, initial decision table can be simplified in following ways:

i. Merge two columns/rows into single if slots in two adjacent rows/columns are the same. *Example:* Two columns Temperature = 18 °C and Temperature = 19 °C are merged into single if all the slots in the adjacent columns [Temperature = 18 °C and Temperature = 19 °C] are the same.
ii. Merge two rows/columns into single if two slots are the same or (if any slot is empty in two adjacent rows/columns and minimum one slot in both the rows/columns is non-empty). *Example*: Two rows Temperature = 40 °C and Temperature = 41 °C, are merged into single.

Based on above mentioned two rules (a and b), membership functions can be rebuilt for simplification process.

v) Derive decision rules from the decision table : To derive a Fuzzy rule (*if-then*), every slot ($Slot_{(w_1,w_2,w_3\ldots\ldots\ldots w_t)} = C_0$) is used in decision table:

Rule 1: **If** $\{A_{11} \wedge/\vee A_{12} \wedge/\vee A_{13} \wedge/\vee \ldots\ldots\ldots\ldots\ldots\ldots \wedge/\vee A_{1n}\}$ **then** C_1

Rule 2: **If** $\{A_{21} \wedge/\vee A_{22} \wedge/\vee A_{23} \wedge/\vee \ldots\ldots\ldots\ldots\ldots\ldots \wedge/\vee A_{2n}\}$ **then** C_2

Rule 3: **If** $\{A_{31} \wedge/\vee A_{32} \wedge/\vee A_{33} \wedge/\vee \ldots\ldots\ldots\ldots\ldots\ldots \wedge/\vee A_{3n}\}$ **then** C_3

. .

. .

Rule m: **If** $\{A_{m1} \wedge/\vee A_{m2} \wedge/\vee A_{m3} \wedge/\vee \ldots\ldots\ldots\ldots\ldots\ldots \wedge/\vee A_{mn}\}$ **then** C_m

Where A_{ij} is an attribute to be retrieved and C_k is an output. Data is gathered from *fuzzy rule base* through various sub processes in fuzzy inference process and based on fuzzy inference rules and their corresponding membership functions, decisions are derived. In the fuzzy logic system, the working of inference engine is similar to reasoning process of human. Crop Name, Temperature, Soil Texture, Season, Pesticide and Fertilizer are antecedents and Productivity is consequent. Several rules are using simultaneously in inference process.

3.2.1.3 Fuzzification

To find the degree of truth for every rule, membership function defined on every input variable is applied to their actual value. We used most popular operator "AND" operator for fuzzy implementation. We used fuzzy "AND" operator to evaluate the rules and produce another variable. Rule is said to be fire if value is non-zero. For every rule, resultant value is used to represent the degree of truth. Apply the truth value of every rule to the output value (productivity) is called inference process. We used "MIN" as an inference rules to calculate the degree of truth through the use of fuzzy logic "AND". Composition process produces the fuzzy subset, which can be further used to convert to single crisp output through Defuzzification.

3.2.1.4 Defuzzification

It is used to convert the value of fuzzy output into crisp output value. We used "MAXIMUM" method in this work for Defuzzification. We select the crisp output for output variable at which fuzzy subset has maximum. Following steps are performed to find a final result from the input given by user through the use of inference process:

1. According to the derived membership functions, numeric input values are transformed into linguistic terms.
2. To determine the output groups, linguistic terms and the decision rules are matching.
3. To form the final decision, Defuzzification of output groups is performed.

For Example: user wants to retrieve the productivity level using Agri-Info.

User Request	**Crop Name**	**Temperature**	**Soil Texture**	**Season**	**Pesticide**	**Fertilizer**	**Productivity**
	Soybean	21-27 ℃	Slity Loam Clay	Winter	Organochlorine	Urea	?

Agri-Info using following rule to find the productivity level Fuzzy Rule based Status Diagnose Module:

Rule: **If** $\{Crop\ Name \wedge Temperature \wedge Soil\ Texture \wedge Season \wedge Pesticide \wedge Fertilizer\}$ **then** $Productivity$

Agri-Info Response	**Crop Name**	**Temperature**	**Soil Texture**	**Season**	**Pesticide**	**Fertilizer**	**Productivity**
	Soybean	21-27 ℃	Slity Loam Clay	Winter	Organochlorine	Urea	**C**

Similarly, any type of request related to different domains can be asked by user and *Agri-Info* executes the user request and send response back to particular user automatically based on the rules defined in *fuzzy rule base*. Through Agri-Info, users can easily diagnosis the agriculture status automatically.

3.2.2 Autonomic Resource Manager

Based on QoS information, resource scheduling is performed using ***S**elf-management of **C**l**O**ud res**O**urces for execu**T**ion of clust**E**red wo**R**kloads (SCOOTER)* technique [17] in this research work. The autonomic component is working based on IBM's autonomic model [18] [19] that considers four steps of autonomic system: 1) Monitor, 2) Analyze, 3) Plan and 4)

Execute and two external interfaces: a) Sensor and b) Effector, as shown in Fig. 1. Autonomic service manager comprises following components:

1) *Sensors* get the information about performance of current state nodes in terms of QoS parameters.
2) *Monitors* are used to collect the information from manager node for monitoring continuously performance variations by comparing expected and actual performance. Expected performance is the threshold value of QoS parameters, which is identified based on their previous history of execution.
3) *Analyze and plan* module start analyzing the information received from monitoring module and make a plan for adequate actions for corresponding alert.
4) *Executor* implements the plan after analysis of current status of system.
5) *Effector* is acting as an interface between nodes to exchange updated information and it is used to transfer the new policies, rules and alerts to other nodes with updated information.

The working of autonomic component is given described in our previous work in detail [17].

4. Graphical User Interface

To validate Agri-Info, we have presented user interaction through a web service i.e. *e-Agriculture* cloud by considering QoS parameters at service level [17]. We have developed this web and mobile application due to following main reasons:

- It is difficult to discuss the technical queries manually, if expert is not present at same office, and managing a meeting personally.
- It is wastage of time to search for and to traverse all the documents listed manually.

4.1 Web Application

Cloud based e-Agriculture web service is global community of practice, where users [(i) agriculture expert, (ii) agriculture officer and (iii) farmer] from all over the world exchange information, ideas, and resources related to the use of ICT for sustainable agriculture for different domains. The e-Agriculture utilizes information, imaging and communication technologies coupled with applications of the Internet for the purpose of providing enhanced agriculture services and information to the farmer and farming community regularly. The e-Agriculture is implemented as centralized system in cloud environment. Several users make use of this system from several geographical locations. Users send their request to the Agri-Info and the Agri-Info gives a response to the request submitted by the user automatically using fuzzy logic-based rules. The various users make use of the system depending upon the rights given to the user. The user's login into the Agri-Info and access the database. This computerized control of the several aspects related to the agriculture to a better management of the different domains such as crops etc. going on in the environment which leads to an improvement in efficiency of work and leads to the customer satisfaction and trust in the organization owing to the timely and proper management. The main objectives to develop a cloud-based e-Agriculture web service are:

- To give answers to the user queries coming from different domains of agriculture from various geographical locations.
- To provide accurate information to users automatically and save time of users.
- To provide a common platform, where multiple farmers can communicate to each other and discuss their problems and any enquiry about any domain.

User interface of cloud-based e-Agriculture web service is shown in Figure 5. The main functions of cloud-based e-Agriculture web service are: searching information using quick search, viewing resource (equipment and tools uses in agriculture) information, get crop status, latest news etc. Figure 6 shows the searching of crop in cloud-based e-Agriculture web service. Figure 7 shows processing of query by cloud-based e-Agriculture web service and results after execution of query. Existing agriculture system manages resources without considering the concept of autonomic and QoS parameters, which leads to performance degradation.

4.2 Mobile Application

We have also developed Android based application for handheld devices for easy access. Main functions of mobile application are: finding productivity, post queries, weather forecasting, getting information about latest equipment and getting alerts about agriculture news and finding crop information as shown in Figure 8. Figure 8 (a) shows the process of finding productivity based on the values of other parameters selected by the user. The value of the productivity is 'high' as shown in Figure 8 (a) while Figure 8 (b) shows the 'low' productivity. Figure 8 (c) shows the process of posting query while Figure 8

(d) shows the posted query with an image. Figure 8 (e) shows the latest information about equipment and latest news alerts are shown in Figure 8 (f). Figure 8 (g) shows the output of weather forecasting and crop information is shown in Figure 8 (h).

Figure 5: User interface of Cloud based e-Agriculture Web Service

5. Performance Evaluation

We have selected CloudSim [16] for simulation and modeling as it supports both system and behavior modeling of cloud system components such as data centers, Virtual Machines (VMs) and resource scheduling policies. CloudSim also supports modeling and simulation of cloud computing environments consisting of both single and inter-networked clouds (federation of Clouds). Moreover, it exposes custom interfaces for implementing policies and scheduling techniques for allocation of VMs under inter-networked cloud computing scenarios. CloudSim has been installed along with its requirements using NetBeans which provide cloud service. 3000 independent cloud workloads (user requests) were generated randomly in CloudSim as Cloudlets. Table 3 shows the characteristics of resources and cloudlets (workloads) that have been used for all the experiments. User cloud workloads are modeled as independent parallel applications are modeled which is compute-intensive. Thus, the data dependency among the cloud workloads in the parallel applications is negligible. Each cloud workload is parallel and is hence considered to be independent of any other cloud workload. Table 4 presents the resource configuration of the simulation. We used three Physical Machines (PMs) with different number of virtual nodes (6, 4 and 2) and virtual nodes are further divided into instances called Execution Components (ECs). Every EC contains their own cost of execution and it is measured with unit (C$/EC time unit (Sec)). EC measures cost per time unit in Cloud dollars (C$).

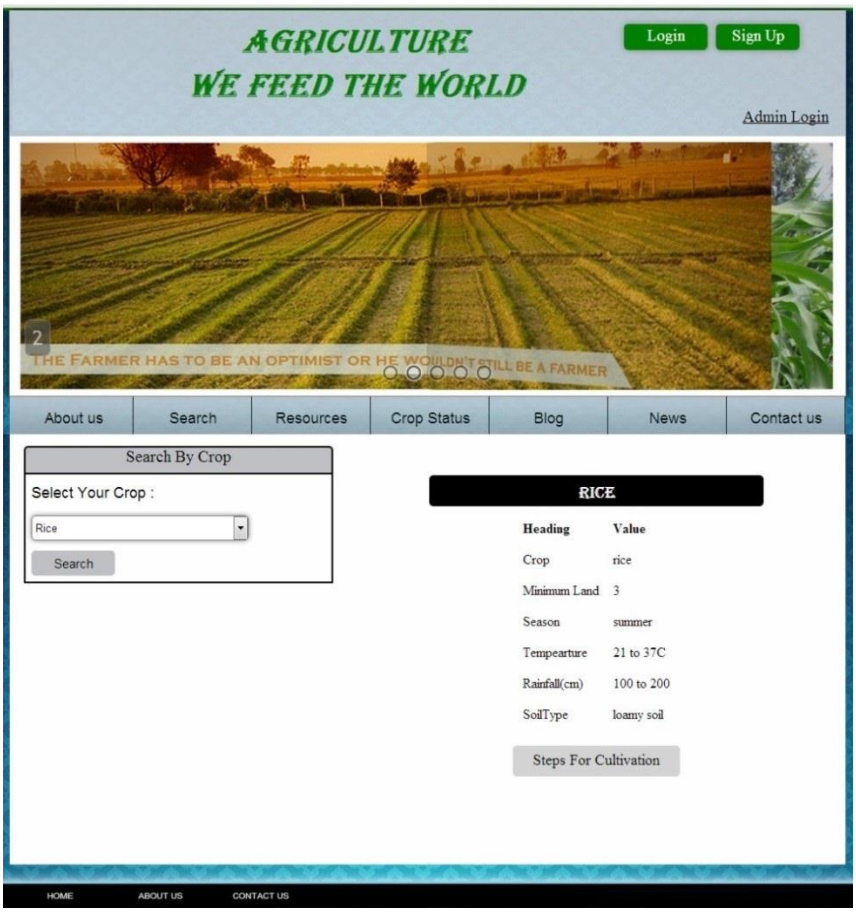

Figure 6: Searching Crop Information in Cloud based e-Agriculture Web Service

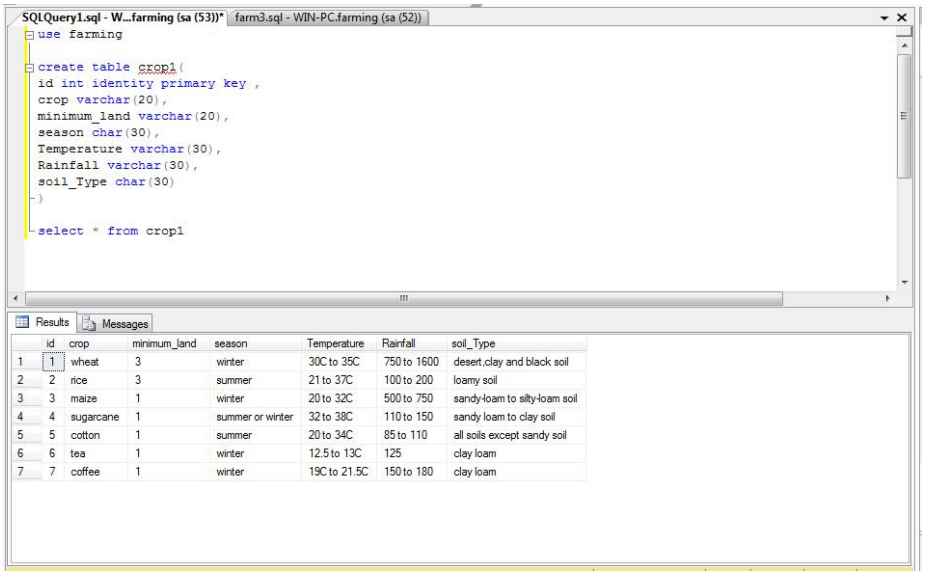

Figure 7: Query Processing and Results in Cloud based e-Agriculture Web Service

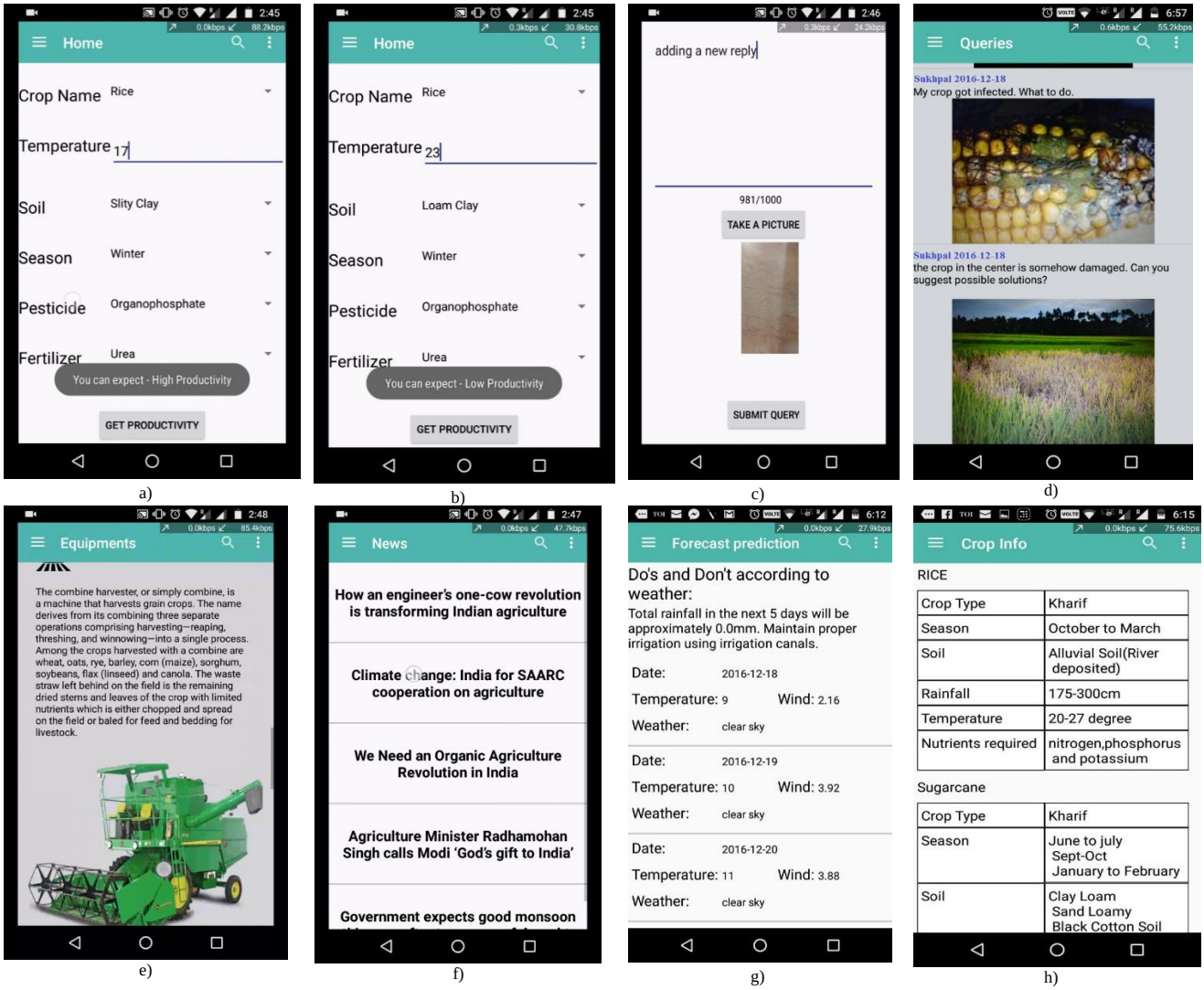

Figure 8: Main functions of Mobile Application: a) Finding Productivity (High), b) Finding Productivity (Low), c) Posting a Query, d) Posted a Query with Image, e) Getting Information about Latest Equipment's, f) Getting Latest News, g) Weather Forecasting and h) Crop Information

Table 3: Scheduling Parameters and their Values

Parameter	Value
Number of resources	250
Number of Cloudlets (Workloads)	3000
Bandwidth	1000 - 3000 B/S
Size of Cloud Workload	10000+ (10%–30%) MB
Number of PEs per machine	1
PE ratings	100-4000 MIPS
Cost per Cloud Workload	$3–$5
Memory Size	2048-12576 MB
File size	300 + (15%–40%) MB
Cloud Workload output size	300 + (15%–50%) MB

Table 4: Configuration Details

Resource_Id	Configuration	Specifications	Core	Operating System	Number of Virtual Nodes	Number of ECs	Price (C$/EC time unit)
R1	Intel Core 2 Duo - 2.4 GHz	6 GB RAM and 320 GB HDD	2	Windows	6 (1 GB and 50 GB)	18	2
R2	Intel Core i5-2310- 2.9GHz	4 GB RAM and 160 GB HDD	2	Linux	4 (1 GB and 40 GB)	12	3
R3	Intel XEON E 52407-2.2 GHz	2 GB RAM and 160 GB HDD	2	Linux	2 (1 GB and 60GB)	6	4

5.1 Experimental Data and Baseline Technique

The application model of Agri-Info is built into CloudSim [16] in order to validate the proposed system through real-time mobile and web application (in other words, data from the experiment is directly fed into the simulator to provide edge-device operational behavior for the resource manager). We have performed the different number of experiments by comparing our proposed QoS-aware Autonomic (Agri-Info) with existing baseline technique i.e. *Smart-Farm* [1], which is *non-autonomic* resource management technique, which neither consider QoS parameter and nor consider autonomic scheduling mechanism.

5.2 Experimental Results

This section presents the experimental results of performance evaluation. Experiment has been conducted with different number of workloads (500-3000) for verification of execution cost, execution time, network bandwidth and latency and to validate Agri-Info as compared to *Smart-Farm*. To verify Agri-Info, we have conducted simulation-based experiments using CloudSim toolkit with the variations in the number of workloads submitted in terms user requests/queries received from web and mobile application. Figure 9 shows the comparisons of Agri-Info and Smart-Farm based on different QoS parameters.

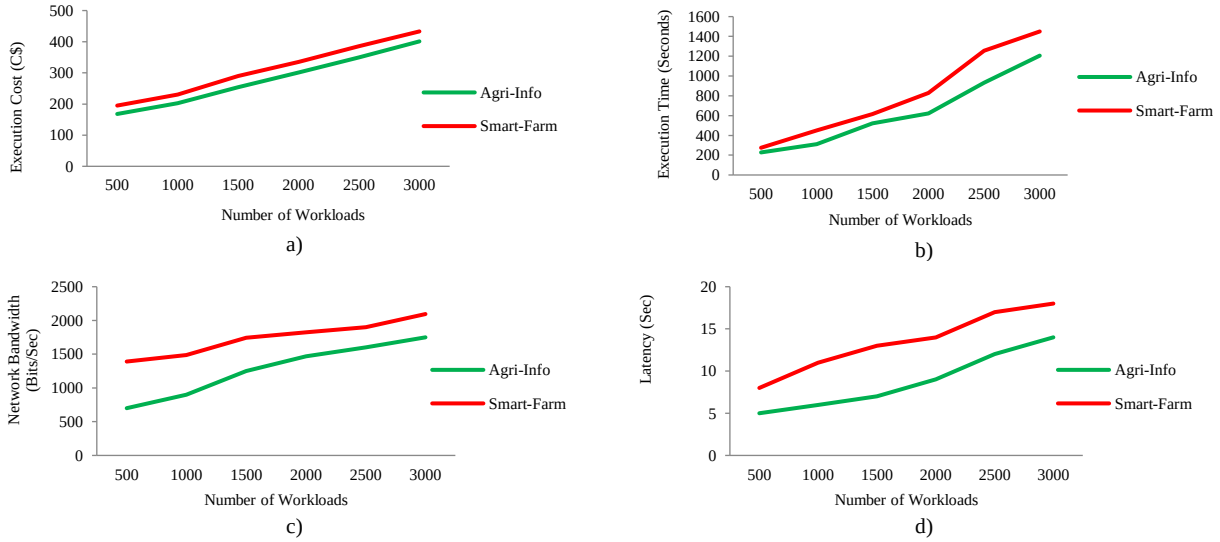

Figure 9: Experimental Results: a) Execution Cost, b) Execution Time, C) Network Bandwidth and d) Latency

Execution cost is defined as the total money that can be spent in one hour to execute the application successfully and execution cost is measured in Cloud Dollars (C$) as mention in Table 4. Figure 9 (a) shows the comparison of Agri-Info and Smart-Farm in terms of execution cost and cost is increasing with increase in number of workloads for Agri-Info and Smart-Farm, but Agri-Info consumes less cost as compared to Smart-Farm. The average value of cost in Agri-Info is 12.46% less than Smart-Farm. In resource scheduler, Agri-Info considers the impact of other workloads on current workload during execution and execute within their specified deadline and budget using STAR [19]. *Execution time* is the amount of time required to execute application successfully and execution time is measured in Seconds. Figure 9 (b) shows the variation of an execution time with different number of workloads and time is increasing with increase in number of workloads for both Agri-Info and Smart-Farm. The average value of execution time in Agri-Info is 10.18% less than Smart-Farm because Agri-Info tracks the resource states automatically for effective decisions using fuzzy logic.

Network Bandwidth is calculated as number of bits transferred/received in a particular workload in one second. Figure 9 (c) shows the value of network bandwidth in Agri-Info is 15.52% less than Smart-Farm. This is because, Agri-Info identifies the network faults automatically using SCOOTER [17] which improves the network bandwidth of Agri-Info as compared to Smart-Farm. *Latency* is a defined as a difference of time of input user request and time of start time of execution of that user request. Figure 9 (d) shows the comparison of Agri-Info and Smart-Farm in terms of latency and its value is increasing with increase in number of workloads for Agri-Info and Smart-Farm, but Agri-Info performs better than Smart-Farm due to automatic execution of resources. The average value of latency in Agri-Info is 13.32% less than Smart-Farm.

5.3 Case Study of Agri-Info in an Indian Village

We have done a pilot study in the Indian Village (Maddoke, Punjab) to check the customer satisfaction level while delivering agriculture as a service using our developed web and mobile application based on Agri-Info. Total 63 farmers have

participated in this case study to test the familiarity and usability of web and mobile application. The Confidence and Fulfillment Matrix based on satisfaction level of farmers is shown in Table 5.

Table 5: Confidence and Fulfillment Matrix

Confidence and Fulfilment Level	Number of Farmers	Application		Customer Satisfaction (%)
		Mobile	Web	
Very Satisfied (100%)	36	31	5	57
Satisfied (75%)	11	10	1	17.5
Neutral (50%)	4	2	2	6.5
Dissatisfied (25%)	5	2	3	8
Completely Dissatisfied (0%)	7	2	5	11

6. Summary and Conclusions

In this paper, QoS-aware cloud based autonomic information system (Agri-Info) for agriculture service has been proposed which manages various types of agriculture related data based on different domains, through different user preconfigured devices. Agri-Info uses autonomic resource manager for efficient resource allocation at infrastructure level after identification of QoS requirements for user's request. We have evaluated the performance of the proposed system in cloud environment using CloudSim toolkit and experimental results show that this system performs better in terms of execution time, cost, network bandwidth and latency. The application model of Agri-Info is built on top of CloudSim in order to validate the proposed system through real-time mobile and web application (in other words, data from the experiment is directly fed into the simulator to provide edge-device operational behavior for the resource manager). The case study of Agri-Info is implemented in an Indian village to evaluate the customer satisfaction amongst farmers. Experimental results show that Agri-Info delivers a superior autonomic solution for farmers and approximate optimum solution for challenges of resource management, using the concept of autonomic computing.

6.1 Future Directions and Open Challenges

Agri-Info can be further enhanced in a larger scope under the following aspects as shown in Figure 10 and discussed below:

1. *Practical implementation of Agri-Info in agriculture industries:* In this research work, the detailed analysis of performance has been conducted through cloud based simulated environment using CloudSim toolkit. The proposed system would be evaluated on real cloud environment in the future, which can be utilized in agriculture industries.
2. *Quality of Service (QoS):* Agri-Info can be extended by developing pluggable scheduler, in which resource scheduling can be changed to incorporate other important QoS parameters such as security, scalability, availability and energy [18].

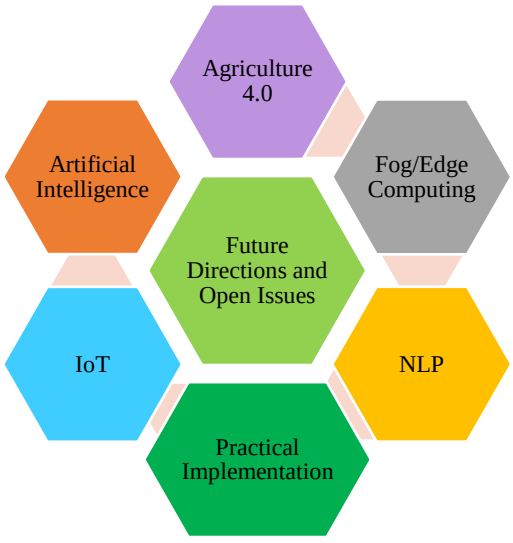

Figure 10: Future Directions and Open Challenges

3. *Natural Language Processing (NLP):* Presently, Agri-Info supports only English language. In future, the other languages can be used to provide the services to end users like farmers.
4. *Artificial Intelligence (AI):* Agriculture enterprises, private companies or cooperatives can utilize Agri-Info by providing agricultural services automatically using AI, which can be helpful to make operational decisions [2]. Knowledge database can be updated automatically based on the new data received by edge or IoT devices.
5. *Agriculture 4.0:* Agriculture 4.0 is a nomenclature for upcoming trends in the Agro-Industry which include a main focus on the IoT, precision agriculture and big data analytics [2]. These technologies will drive greater business efficiencies in the face of increasing populations and climate change. Further, Agri-Info can be used to analyze the influence of Industry 4.0 in the Agro-Industry, transportation sector, crop rotation and landscape management.
6. *Fog/Edge Computing:* Proposed service can interact with sensor data or IoT devices using Fog/Edge Computing platform to reduce latency and response time of service [2]. New data aggregation mechanism can be designed to collect data from heterogenous IoT/Edge devices.

Declaration of Competing Interest

We do not have any conflicts of interest.

Acknowledgments

This research work is an updated version of Technical Report CLOUDS-TR-2015-2, Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, Nov. 27, 2015 (Available at https://arxiv.org/abs/1511.08986) and summarized and extended version of research article titled "IoT Based Agriculture as a Cloud and Big Data Service: The Beginning of Digital India" [12], published in *Journal of Organizational and End User Computing*, 2017 (Available at https://doi.org/10.4018/JOEUC.2017100101). We would like to thank the editor, area editor and anonymous reviewers for their valuable comments and suggestions to help and improve our research paper. We would like to thank Manmeet Singh (Scientist at Indian Institute of Tropical Meteorology, India) for his useful suggestions and discussion to improve the quality of the paper.

References

[1] Muangprathub, Jirapond, Nathaphon Boonnam, Siriwan Kajornkasirat, Narongsak Lekbangpong, Apirat Wanichsombat, and Pichetwut Nillaor. "IoT and agriculture data analysis for smart farm." Computers and electronics in agriculture 156 (2019): 467-474.
[2] Sukhpal Singh Gill et al. "Transformative Effects of IoT, Blockchain and Artificial Intelligence on Cloud Computing: Evolution, Vision, Trends and Open Challenges." Internet of Things (2019), vol. 8.
[3] Yan, Hui, Ping Yu, and Duo Long. "Study on Deep Unsupervised Learning Optimization Algorithm Based on Cloud Computing." In 2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), pp. 679-681. IEEE, 2019.
[4] Jinbo, Chen, Zhong Yu, and Anthony Lam. "Research on monitoring platform of agricultural product circulation efficiency supported by cloud computing." Wireless Personal Communications 102, no. 4 (2018): 3573-3587.
[5] Zamora-Izquierdo, Miguel A., José Santa, Juan A. Martínez, Vicente Martínez, and Antonio F. Skarmeta. "Smart farming IoT platform based on edge and cloud computing." Biosystems engineering 177 (2019): 4-17.
[6] Kaloxylos, Alexandros, Robert Eigenmann, Frederick Teye, Zoi Politopoulou, Sjaak Wolfert, Claudia Shrank, Markus Dillinger, "Farm management systems and the Future Internet era", *Computers and Electronics in Agriculture*, 89 (2012): 130-144.
[7] Ranya Elsheikh, Abdul Rashid B. Mohamed Shariff, Fazel Amiri, Noordin B. Ahmad, Siva Kumar Balasundram, and Mohd Amin Mohd Soom, "Agriculture Land Suitability Evaluator (ALSE): A decision and planning support tool for tropical and subtropical crops", *Computers and Electronics in Agriculture*, 93 (2013): 98-110.
[8] Sørensen, C. G., S. Fountas, E. Nash, Liisa Pesonen, Dionysis Bochtis, Søren Marcus Pedersen, B. Basso, and S. B. Blackmore. "Conceptual model of a future farm management information system." Computers and electronics in agriculture 72, no. 1 (2010): 37-47.
[9] Sørensen, C. G., Liisa Pesonen, D. D. Bochtis, S. G. Vougioukas, and Pasi Suomi. "Functional requirements for a future farm management information system." Computers and Electronics in Agriculture 76, no. 2 (2011): 266-276.
[10] Shitala Prasad, Sateesh K. Peddoju, and Debashis Ghosh, "AgroMobile: A Cloud-Based Framework for Agriculturists on Mobile Platform", *International Journal of Advanced Science and Technology*, 59 (2013): 41-52.
[11] Jeong, Seokkyun, Hoseok Jeong, Haengkon Kim, and Hyun Yoe. "Cloud computing based livestock monitoring and disease forecasting system." International Journal of Smart Home 7, no. 6 (2013): 313-320.
[12] Gill, Sukhpal Singh, Inderveer Chana, and Rajkumar Buyya. "IoT based agriculture as a cloud and big data service: the beginning of digital India." Journal of Organizational and End User Computing (JOEUC) 29, no. 4 (2017): 1-23.
[13] Abraham, Ajith. "Rule-Based Expert Systems." Handbook of measuring system design (2005).
[14] Fahmy, M. M. M. "A fuzzy algorithm for scheduling non-periodic jobs on soft real-time single processor system." Ain Shams Engineering Journal 1, no. 1 (2010): 31-38.
[15] Tzung-Pei Hong, and Chai-Ying Lee, "Induction of fuzzy rules and membership functions from training examples", *Fuzzy sets and Systems* 84(1) (1996): 33-47.
[16] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", Software: Practice and Experience, 41(1): 23-50, Wiley Press, New York, USA, January 2011.

[17] Gill, Sukhpal Singh, and Rajkumar Buyya. "Resource provisioning based scheduling framework for execution of heterogeneous and clustered workloads in clouds: from fundamental to autonomic offering." Journal of Grid Computing 17, no. 3 (2019): 385-417.
[18] S.S. Gill, I. Chana, M. Singh, R. Buyya RADAR: self-configuring and self-healing in resource management for enhancing quality of cloud services Concurr. Comput. Pract. Exp., 31 (1) (2019), p. e4834.
[19] Singh, Sukhpal, Inderveer Chana, and Rajkumar Buyya. "STAR: SLA-aware autonomic management of cloud resources." IEEE Transactions on Cloud Computing (2017). https://doi.org/10.1109/TCC.2017.2648788
[20] Rok Rupnik, Matjaž Kukar, Petar Vračar, Domen Košir, Darko Pevec, Zoran Bosnić, AgroDSS: A decision support system for agriculture and farming, Computers and Electronics in Agriculture, Volume 161, 2019, Pages 260-271
[21] Yiannis Ampatzidis, Li Tan, Ronald Haley, Matthew D. Whiting, Cloud-based harvest management information system for hand-harvested specialty crops, Computers and Electronics in Agriculture, Volume 122, 2016, Pages 161-167
[22] Alejandro Belanche, A. Ignacio Martín-García, Javier Fernández-Álvarez, Javier Pleguezuelos, Ángel R. Mantecón, David R. Yáñez-Ruiz, Optimizing management of dairy goat farms through individual animal data interpretation: A case study of smart farming in Spain, Agricultural Systems, Volume 173, 2019, Pages 27-38
[23] Madalin Colezea, George Musat, Florin Pop, Catalin Negru, Alexandru Dumitrascu, Mariana Mocanu, CLUeFARM: Integrated web-service platform for smart farms, Computers and Electronics in Agriculture, Volume 154, 2018, Pages 134-154
[24] Sjaak Wolfert, Lan Ge, Cor Verdouw, Marc-Jeroen Bogaardt, Big Data in Smart Farming - A review, Agricultural Systems, Volume 153, 2017, Pages 69-80
[25] Raimo Nikkilä, Ilkka Seilonen, and Kari Koskinen, "Software architecture for farm management information systems in precision agriculture", Computers and Electronics in Agriculture, 70(2) (2010): 328-336.