Journal of Organizational and End User Computing
Volume 29 • Issue 4 • October-December 2017

IoT Based Agriculture as a Cloud and Big Data Service:
The Beginning of Digital India

Sukhpal Singh Gill, CLOUDS Lab, School of Computing and Information Systems, University of Melbourne, Melbourne, Australia
Inderveer Chana, Computer Science and Engineering Department, Thapar University, Patiala, India
Rajkumar Buyya, CLOUDS Lab, School of Computing and Information Systems, University of Melbourne, Melbourne, Australia

ABSTRACT

Cloud computing has transpired as a new model for managing and delivering applications as services efficiently. Convergence of cloud computing with technologies such as wireless sensor networking, Internet of Things (IoT) and Big Data analytics offers new applications' of cloud services. This paper proposes a cloud-based autonomic information system for delivering Agriculture-as-a-Service (AaaS) through the use of cloud and big data technologies. The proposed system gathers information from various users through preconfigured devices and IoT sensors and processes it in cloud using big data analytics and provides the required information to users automatically. The performance of the proposed system has been evaluated in Cloud environment and experimental results show that the proposed system offers better service and the Quality of Service (QoS) is also better in terms of QoS parameters.

KEYWORDS

Agriculture as a Service, Autonomic Management, Big Data, Cloud Computing, Internet of Things

1. INTRODUCTION

Emergence of ICT (Information and Communication Technologies) plays an important role in the agriculture sector by providing services through computer-based agriculture systems (Singh and Chana, 2015). But these agriculture systems are not able to fulfill the needs of today's generation due to processing of large amount of data, lack of important requirements like processing speed, data storage space, reliability, availability, scalability etc. and even resources used in computer-based agriculture systems are not utilized efficiently. Agriculture-as-a-Service (AaaS) applications exhibit Big data characteristics. For example, the volume of agriculture dataset captured by environments such as Open Government Data Platform India (data.gov.in, 2015), India Agriculture and Climate Data Set (Sanghi et al.), and regional land and climate modelling in China (Shangguan et al., 2012) can be in order of 1000000 records with size of 3.5 GB. The data is coming in large data variety and volume from both users in the form of images like damaged crop images due to weather, insects etc. and devices through Internet of Things (IoT) sensors and satellites (GPS systems) that send weather related images. As a result of regular capturing and collection of datasets, they grow with the velocity

DOI: 10.4018/JOEUC.2017100101

of 80.72 KB/minute or more (data.gov.in, 2015). To solve the problem of existing agriculture systems, there is a need to develop a cloud-based service 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 sensors through preconfigured devices, ii) classify the gathered data (heterogeneous, high volume of big data) into various classes through analysis, iii) store the classified information in cloud repository for future use, and iv) automatic diagnosis of the agriculture status. As large number of users are using agriculture systems operating on large datasets simultaneously, there is a need of highly scalable and elastic distributed computing environment such as cloud computing. In addition, cloud-based autonomic information system should be able to identify the QoS (Quality of Service) requirements of user request and resources should be allocated efficiently to execute the user request based on these requirements.

The main aim of this paper is to design architecture of Agriculture-as-a-Service (AaaS) that manages various types of agriculture-related data based on different domains. This is realized through the following objectives: i) propose an autonomic resource management technique which is used to a) gather the information from various users through preconfigured devices, IoT sensors, GPS (Global Positioning System), etc. b) extract the attributes, c) analyze the information by creating various classes based on the information received, d) store the classified information in cloud repository for future use and e) diagnose the agriculture status automatically and ii) perform resource allocation automatically at infrastructure level after identification of QoS requirements of user request.

The rest of the paper is organized as follows. Section 2 presents related work of existing agricultures systems. Proposed architecture is presented in Section 3. Section 4 presents Autonomic Resource Management. Sections 5 describe the experimental setup and present the results of evaluation. Section 6 presents conclusions and future scope.

2. RELATED WORK

Existing research reported that few agriculture systems have been developed with limited functionality. Related work of existing agriculture systems has been presented in this section.

2.1. Existing Agriculture Systems

Ranya et al. (2013) presented ALSE (Agriculture Land Suitability Evaluator) to study various types of land to find the appropriate land for different types of crops by analyzing geo-environmental factors. ALSE used GIS (Global Information System) capabilities to evaluate land using local environment conditions through digital map and based on this information decisions can be made. Raimo et al. (2010) proposed FMIS (Farm Management Information System) 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. (2010) 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. Zhao (2002) presented an analysis of web-based agricultural information systems and identified various challenges and issues still pending in these systems. Due to lack of automation in existing agriculture system, the system is taking longer time and is difficult to handle dynamic needs of user which leads to customer dissatisfaction. Sorensen et al. (2011) 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. Yuegao et al. (2004) proposed WASS (Web-based Agricultural Support System) and identified functionalities (information, collaborative work and decision support) and characteristics of WASS. Based on characteristics, authors divided WASS into three subsystems: production, research-education and management.

Reddy at el. (1995) proposed GIS based DSS (Decision Support System) framework in which Spatial DDS has been designed for watershed management and management of crop productivity at regional and farm level. GIS is used to gather and analyze the graphical images for making new rules and decisions for effective management of data. Shitala et al. (2013) 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. (2013) 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. The proposed QoS-aware Cloud Based Autonomic Information System (AaaS) has been compared with existing agriculture systems as described in Table 1.

All the above research works have focused on different domains of agriculture with different QoS parameters. None of the existing agriculture systems considers self-management of resources. Due to lack of automation of resource management, services become inefficient which further leads to customer dissatisfaction. The proposed system is a novel QoS-aware cloud based autonomic information system and considers various domains of agriculture and, allocates and manages the resources automatically which is not considered in other existing agriculture systems.

3. AGRICULTURE-AS-A-SERVICE ARCHITECTURE

The existing agriculture systems are not able to fulfill the needs of today's generation due to lacking in important requirements like processing speed, data storage space, reliability, availability, scalability etc. Even 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 that delivers Agriculture-as-a-Service. This section presents architecture of cloud-based autonomic information system for agriculture service called *AaaS* that manages various

Table 1. Comparisons of existing agriculture systems with proposed system (AaaS)

Agriculture System	Mechanism	QoS-aware (Parameter)	Domains	Data Classification	Resource Management	Big Data
ALSE (Elsheikh et al., 2013)	Non-Autonomic	**Yes** (Suitability)	Soil	Yes	No	No
FMIS (Nikkila et al., 2010)	Non-Autonomic	**No**	Pest and Crop	No	No	No
WASS (Hu et al., 2004)	Non-Autonomic	**No**	Productivity	No	No	No
AgroMobile (Prasad et al., 2013)	Non-Autonomic	**Yes** (Data accuracy)	Crop	Yes	No	No
DFLMS (Jeong et al., 2013)	Non-Autonomic	**No**	Crop	No	Yes	No
Proposed System (AaaS)	Autonomic	**Yes** (Cost, Time, Resource Utilization, Latency, Throughput and Attack Detection Rate)	Crop, Weather, Soil, Pest, Fertilizer and Irrigation	Yes	Yes	Yes

types of agriculture-related data based on different domains. Architecture of AaaS is shown in Figure 1. QoS parameters (execution time and cost) must be identified before the allocation of resources. AaaS is the key mechanism that ensures that the resource manager can serve large amount of requests without violating SLA terms and dynamically manages the resources based on QoS requirements identified by QoS manager. The services of AaaS has been divided into three types: SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service). In SaaS, a user interface is designed in which users can interact with system. Aneka is a .NET-based application development PaaS, which is used as a scalable cloud middleware to make interaction between cloud subsystem and user subsystem. In IaaS, an autonomic resource manager manages the resource automatically based on the identified QoS requirements of a particular request. The architecture of AaaS comprises of two subsystems: i) user and ii) cloud.

3.1. User Subsystem

This subsystem provides a user interface, in which different type of users interact with AaaS to provide and get useful information about agriculture based on different domains. Nine types of information of different domains in agriculture has been considered: 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. The agriculture expert shares professional knowledge by answering farmer queries and updates the AaaS database based on the latest research done in the field of agriculture with respect to their domain. Agriculture officers are the government officials that provide the latest information about new agriculture policies, schemes, and rules passed by the government. Farmer is an important entity of AaaS who can take maximum advantage by asking his queries and getting automatic reply after analysis. 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 AaaS. The queries received from user(s) are forwarded to cloud repository for updates and response sends back to particular user on their preconfigured devices (tablets, mobile phones, laptops etc.) via internet.

3.2. Cloud Subsystem

This subsystem contains the platform in which agriculture service is hosted on a cloud. Details about users and agriculture information are stored in a cloud repository in different classes for different domains with unique identification number. The information is monitored, analyzed, and processed continuously by AaaS. The analysis process consists of various sub processes: selection, data preprocessing, transformation, classification and interpretation as shown in Figure 1. Different classes for every domain and sub classes for further categorization of information have been designed. In storage repository, 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 preconfigured devices. Further, a number of users can use cloud-based agriculture service so the QoS manager and autonomic resource manager in cloud subsystem have been integrated. QoS manager identifies the QoS requirements based on the number and type of user queries as discussed in previous research work (Jeong et al., 2013; Singh and Chana, 2015; Singh et al., 2015). Based on QoS requirements, autonomic resource manager identifies resource requirements automatically and allocates and executes the resources at infrastructure level. Performance monitor is used to verify the performance of system and also maintain it automatically. If the system will not be able to handle the request automatically then the system generates an alert.

3.2.1. Cloud-Based Agriculture Service

Cloud-based agriculture service provides a user platform through which user can access agriculture service as shown in Figure 2. Firstly, agriculture service allows user to create profile for interaction with AaaS. After profile creation, the user is required to provide his personal details along with the

Figure 1. Agriculture-as-a-Service architecture

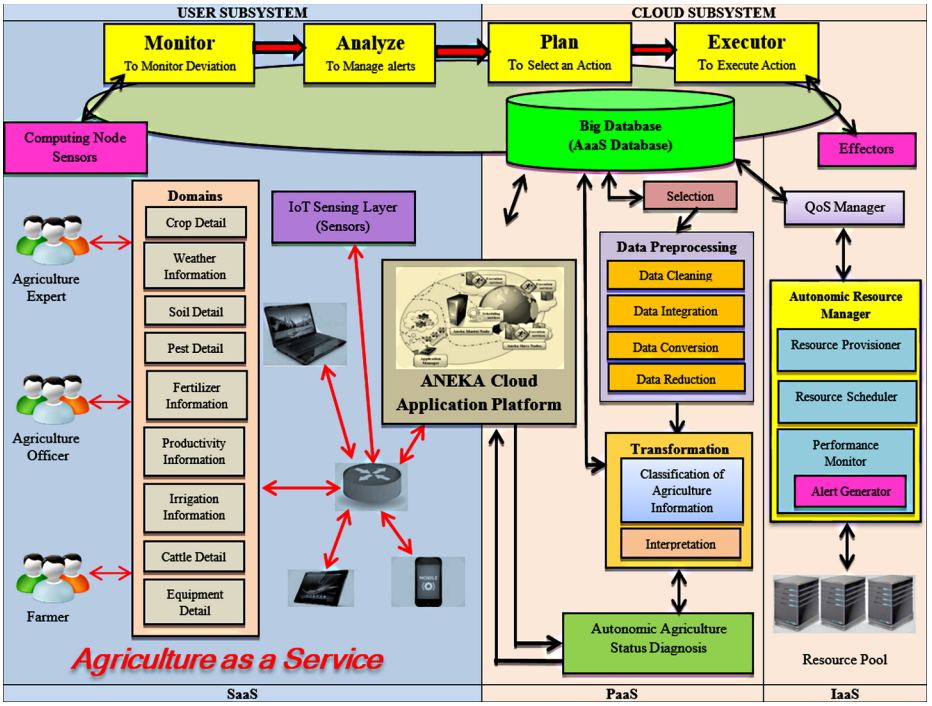
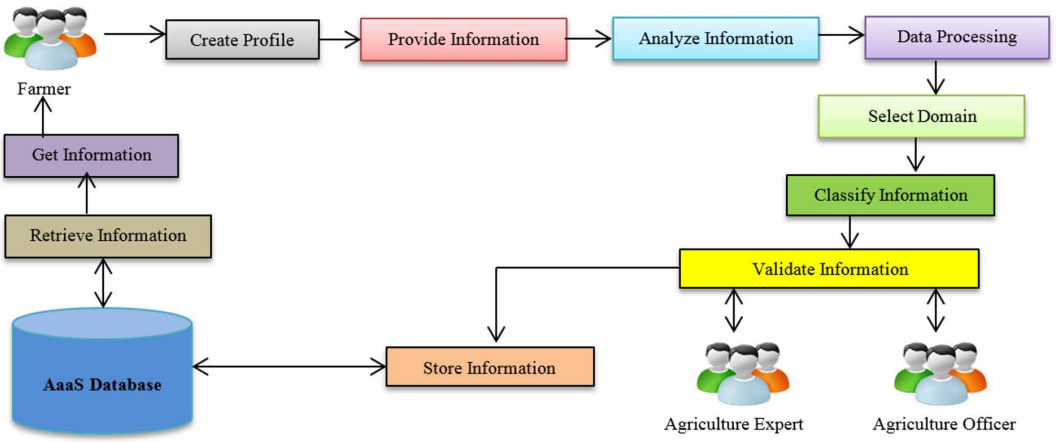


Figure 2. Functional aspects of AaaS



details of information domain. AaaS analyses the information to verify whether the data is complete or not for further processing by performing various checks. Further data is processed and redundancy of data is removed and data is used to select domain to which data belongs. Information is classified properly in order with unique identification number. This information is forwarded to agriculture experts and agriculture officers for final validation through preconfigured devices. After successful validation of information, it is stored in *AaaS* database. If user wants to know the response of their query, then system will automatically diagnose the user query and send the response back to that user.

3.2.2. Detailed Methodology

AaaS allows users to upload the data related to different domains of agriculture through preconfigured devices and classified them based on the domains specified in database. Subtasks of information gathering and provided in AaaS are: i) selection, ii) preprocessing, iii) transformation, iv) classification and v) interpretation. In *selection*, target datasets are created based on the relevant information that will further be considered for analysis in next sub process. In *preprocessing*, different users have different information regarding agriculture. To develop a final training set, there is need of preprocessing steps because data might contain some missing sample or noise components. In AaaS, data preprocessing contains four different sub processes: i) data cleaning, ii) data integration, iii) data conversion and iv) data reduction. Data *transformation* provides an interface between data analysis sub process (classification) and data preprocessing. After data preprocessing, this process converts the labeled data into adequate format suitable for classification. In *classification*, AaaS classify the agriculture information of different users of different domains based on the extracted data. K-NN (k-Nearest Neighbor) classification mechanism has been used in this research work to identify the different class labels of users. K-NN is supervised machine learning technique which is used to classify the unknown data using training data set generated by it. K-NN used to identify the productivity level through Training Instance Dataset (TID). Figure 3 describes the K-NN Algorithm.

In K-NN algorithm, distance is computed from one specific instance to every training instance to classify that unknown instance. Both k-nearest neighbor and *k* minimum distance is determined and output class label is identified among *k* classes. During training phase, K-NN Algorithm utilizes training data. Figure 4 illustrates the classification process used in this research work.

K-NN model is used to identify the productivity level through Training Instance Dataset (TID). Five levels of productivity (A - E) have been fixed as shown in Table 2. The level *'A'* indicates the productivity is very high while level *'E'* indicates the productivity is very low. Based on the given information, TID identifies the class in which given data belongs.

Test data is an input of this model and it is compared with TID and identifies the class in which data laid using following rule:

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

The final step is to *interpret* the agriculture data submitted by different users of different domains which helps user to understand the classified datasets. AaaS is capable to diagnose the agriculture status based on the information entered by user and send the diagnosed agriculture status to particular user

Figure 3. Pseudo code of K-NN algorithm

Algorithm 1: K-NN Algorithm

Learning Phase:
To create the Training Instance Dataset (TID)
Classification Phase:
for every unknown instance dp_a

1. Identify dp_1, dp_2, ... dp_k, the k most nearest instances from TID, where dp_1, dp_2, ... dp_k, are *data points* $\in$ *region (r).*
2. Set class label (cl) is equal to most repeated cl of k nearest instances.
3. Return cl

end for

Figure 4. Classification process

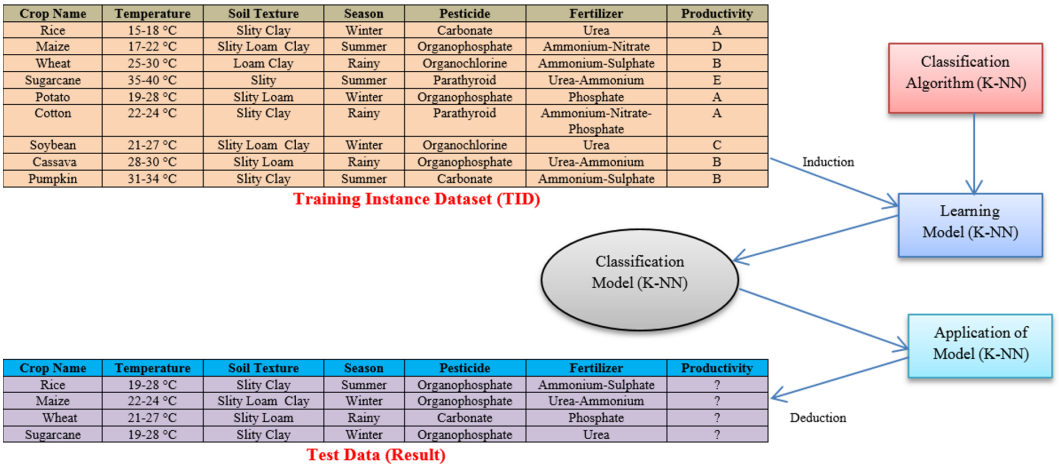


Table 2. Productivity Levels

Productivity Level	Description
A	Very High Productivity
B	High Productivity
C	Neutral Productivity
D	Low Productivity
E	Very Low Productivity

automatically. Six attributes have been considered: Crop Name, Temperature, Soil Texture, Season, Pesticide and Fertilizer and one output: Productivity. Based on these six attributes, AaaS designs rules. Values for six variables are considered as TID. For example, refer to Table 3.

AaaS uses the rule shown in Table 3 to find the productivity level using TID (see Table 4).

Similarly, any type of query related to different domains can be asked by users and AaaS executes the user query and send response back to particular user automatically based on the rules defined in *AaaS* database. Through AaaS, users can easily diagnose the agriculture status automatically.

3.2.3. Infrastructure Management (IaaS)

Efficient management of infrastructure in cloud is mandatory to maintain the performance of the Agri-Info. It comprises of two sub units: QoS Manager and Resource Manager.

Table 3. User wants to retrieve the productivity level using AaaS

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

Table 4. AaaS response utilized to in order to find the productivity level using TID

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

3.2.3.1. QoS Manager

User submits a request to Agri-Info to retrieve some specific agriculture related information. Agri-Info identifies the QoS parameters required to process the user request through analysis based on user request. Based on the key QoS requirements of a particular user request, the *QoS Manager* puts the user request into critical and non-critical queues through QoS assessment. For QoS assessment, *QoS Manager* will calculate the execution time of user request and find the approximate user request completion time. If the completion time is lesser than the desired deadline then it will execute immediately with the available resources and release the resource(s) back to resource manager for another execution otherwise calculate extra number of resources required and provide from the reserved stock for current execution.

3.2.3.2. Resource Manager

Further, two resource scheduling policies (Singh and Chana, 2015) are used to schedule the resources for execution of user queries: time based and cost based scheduling policy. *Time based scheduling policy* works as per following: First, the allocation agent begins to compute the Deadline Time of the user request in the given budget. Allocate resources based on time, the user request which has shortest Deadline Time will execute first. If the two requests have same deadline time then that request will execute first that has lesser execution time. The allocation agent then schedules all the requests with smallest execution time request to the resources that provide high QoS. The rules for time based scheduling policy are described in Table 5 along with their conditions.

Cost based scheduling policy works as per following: First, the allocation agent begins to compute the cost of each request then sort, as the priority is given to the request which has maximum budget. If the two requests have same budget then that request will execute first that has lesser execution time. The allocation agent then schedules all the requests with high budget request to the resources that provide high QoS. Finally, all other requests are scheduled on the available resources set. The rules for cost based scheduling policy are described in Table 6 along with their conditions.

4. AUTONOMIC RESOURCE MANAGEMENT

Working of autonomic element of Agri-Info is based on IBM's autonomic model that considers four steps of autonomic system: i) monitor, ii) analyze, iii) plan and iv) execute as shown in Figure 1. The

Table 5. Rules of time based resource scheduling

Request Pending	Urgency	Add Resource	Request
Yes	Yes	Reserve	Submit
Yes	No	Available	Submit
No	-	-	Finish

Table 6. Rules of cost based resource scheduling

Request Pending	$R_A > 0$	$E_t > W_d$	$B_A > P_r$	Status
Yes	True	True	True	Add Resource
Yes	False	True	True	Add Resource
No	-	-	-	Finish
Yes	True	False	True	Finish
Yes	True	True	False	Finish

R_A = Resource Available, E_t = Estimated Time, P_r = Resource Price, W_d = Desired Deadline and B_A = Available Budget. Details of both time and cost based scheduling policy is given in previous research work (Singh and Chana, 2015).

objective of resource provisioning in autonomic resource management is to provision the resources to process user requests. The requests submitted should be executed within their budget and deadline. Requests submitted by user to resource provisioner are stored as bulk of workloads for their execution. All the submitted workloads are analyzed based on their QoS requirements. Based on importance of the attribute, weights for every cloud workload are calculated. After that, workloads are clustered based on k-means based clustering algorithm for better resource provisioning (Singh et al., 2015). If the value of workloads executes within deadline and budget and [Resource Consumption and Requests Missed is lesser than Threshold Value] then it will provision resources otherwise generate alert for analyses the workload again.

After successful provisioning of resources, Resource Scheduler (RS) takes the information from the appropriate workload after analyzing the various workload details which user request demanded (Singh and Chana, 2015). Knowledge Base contains details of all the resources available in resource pool and reserve resource pool. Based on Cloud consumer details, RS assigns resources and executes Cloud workloads. During execution of a particular cloud workload, the Resource Executor (RE) will check the current workload. If the resources are sufficient for execution then it will continue with execution otherwise request for more resources. If the value of Resource Consumption and Requests Missed is lesser than threshold value, then RE will execute workloads otherwise RE will generate alert. After successful execution of Cloud workloads, RE releases the free resources to resource pool and RE is ready for execution of new cloud workloads. During execution of user requests, performance is monitored continuously using sub unit performance monitor to maintain the efficiency of Agri-Info and generates alert in case of performance degradation. Alerts can be generated in two conditions generally: i) if resource consumption is more than threshold values of resource consumption to execute user request (*Action:* Reallocates resources) and ii) if the number of missed requests are greater than the threshold value (*Action*: Predict QoS Requirements Again). Same action is performed twice, if Agri-Info fails to correct it then system will be treated as down. Components of autonomic system are described below:

4.1. Sensors

Sensors get the information about performance of other nodes using in the system and their current state. Firstly, the updated information from processing nodes is transfer to manager node then manager node transfers this information to sensors. Updated information includes information about QoS parameters (execution time, execution cost and resource utilization etc.).

4.2. Monitor

Initially, Monitors are used to collect the information from sensors for monitoring continuously performance variations by comparing expected and actual performance, and monitors the value of

resource consumption and missed requests. Actual information about performance is observed based QoS parameters and transfers this information to next module for further analysis.

4.3. Analysis and Plan

Analyze and plan module start analyzing the information received from monitoring module and make a plan for adequate actions for corresponding alert. Following formula is used to calculate Resource Consumption (Equation 1):

$$Resource_{Consumption} = \sum_{i=1}^{n} \left(\frac{\text{Actual Resource Usage}}{\text{Predicted Resource Usage}} \right) \tag{1}$$

where $\text{Actual Resource Usage}$ is usage of resource to execute particular number of user requests and $\text{Predicted Resource Usage}$ is resource usage estimated before actual execution and *n* is the number of resources. Assumed: $\left[\text{Predicted Resource Usage} \leq \text{Actual Resource Usage}\right]$. Value of $ResourceConsumption$. is more than 1 generally because $\text{Actual Resource Usage}$ is more than $\text{Predicted Resource Usage}$ but ideally it will be 1 when both are equal. In this research rk, maximum values for $ResourceConsumption$ has been fixed and that is called threshold value. Following formula is used to calculate number of requests missed $\left(Requests_{Missed}\right)$ in a particular period of time (Equation 2):

$$Requests_{Missed} = [Number\ of\ Requests\ Executed\ Successfully - Number\ of\ Requests\ Missed\ Deadline] \tag{2}$$

For successful execution of resources, value of $Requests_{Missed}$ is lesser than threshold value. Algorithm 1 is used to analyses the performance of management of resources.

With the help of (Equation 1) and (Equation 2), resource consumption is calculated and allocates the resources for execution and then compares the resource consumption with threshold value $\left(Th_c\right)$. If resource consumption is less than threshold value and value of $Requests_{Missed}$ is less than threshold value $\left(Th_m\right)$ then execution of resources continues otherwise no resource is allocated and process of reallocation is started using Algorithm 1. After meeting this condition, resources are allocated for

Algorithm 1. Analyzing and Panning Unit (AU)

```
1.  # AUTONOMIC ANALYZING AND PANNING
2.  if (Required Resources > Provided Resources) then
    Allocate new resource from reserve resource pool with minimum Resource_consumption and Requests_Missed and [ Resource_consumption ≤ Th_c
    && Requests_Missed ≤ Th_m]
3.  end if
4.  for all resources
5.    if ( Resource_consumption ≤ Th_c && Requests_Missed ≤ Th_m) = = 'FALSE' then
6.        Restart the resource and start execution
7.    else if
8.        Declared resource as dead node and removed
9.        Resource(s) is required to process data without degradation in performance
10.       if (Availability of Resource(s) = = 'TRUE')
          Allocate new resource from resource pool with minimum Resource_consumption and Requests_Missed and [ Resource_consumption ≤
          Th_c && Requests_Missed ≤ Th_m]
          else
11.       Resource(s) is required to process data without degradation in performance
          Add new resource from reserve resource pool with minimum Resource_consumption and Requests_Missed and
          [ Resource_consumption ≤ Th_c && Requests_Missed ≤ Th_m]
12.   end if
13. end for
```

further execution and value of resource consumption and $Requests_{Missed}$ are checked periodically. In case of more value than threshold, alert will be generated by performance monitor.

4.4. Executor

Executor implements the plan after analyzing completely. To reduce the execution time and execution cost and improve resource utilization is a main objective of executor. Based on the output given by analysis and executor tracks the new user request submission and resource addition, and take the action according to rules described in knowledge base.

4.5. Effector

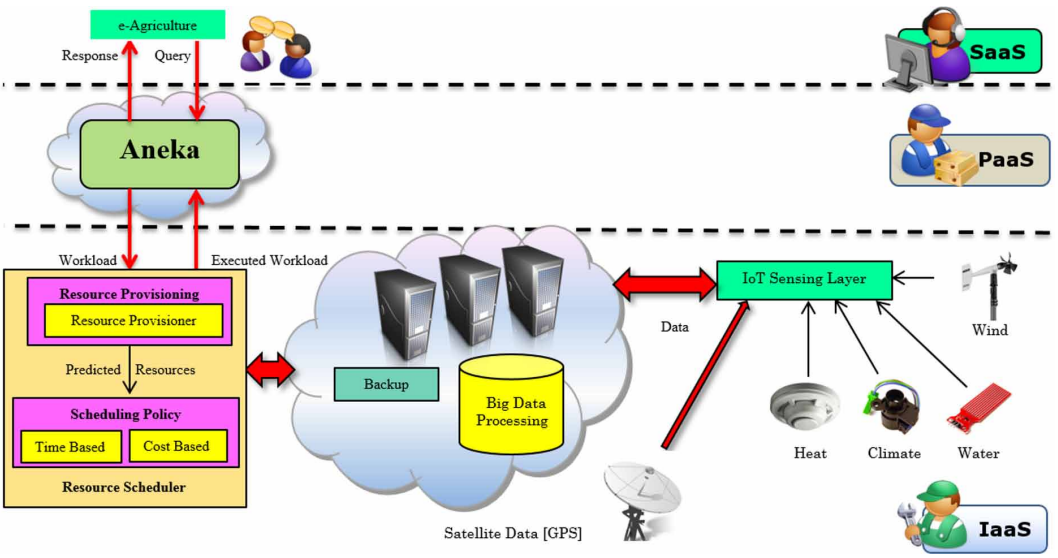
Effector is used to exchange updated information and it is used to transfer the new policies, rules and alerts to other nodes with updated information.

5. PERFORMANCE EVALUATION

The aim of this performance evaluation is to demonstrate that it is feasible to implement and deploy the agriculture as a service on real cloud resources. Tools used for setting up cloud environment for performance analysis are Microsoft Visual Studio 2010 (SaaS), Aneka (PaaS), SQL Server 2008, and Citrix Xen Server (IaaS). Aneka has been installed along with its requirements on all the nodes that provide cloud service. Nodes in this system can be added or removed based on the requirement. AaaS is installed on main server and tested on virtual cloud environment that has been established at *CLOUDS Lab, University of Melbourne, Australia*. Different number of virtual machines have been installed on different servers, and deployed the AaaS to measure the variations. In this experimental setup, three different cloud platforms are used: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) as shown in Figure 5.

At ***SaaS level***, Microsoft Visual Studio is used to develop e-agriculture web service to provide user interface in which user can access service from any geographical location. At ***PaaS level***, Aneka cloud application platform is used as a scalable cloud middleware to make interaction between IaaS

Figure 5. Deployment of components at runtime and their interaction



and SaaS, and continually monitor the performance of the system. At ***IaaS level***, three different servers (consist of virtual nodes) have been created through Citrix Xen Server and SQL Server has been used for data storage. Scheduler as shown in Figure 5, runs at IaaS level on Citrix Xen Server. Computing nodes used in this experiment work are further categorized into three categories as shown in Table 7. The execution cost is calculated based on user request and deadline (if deadline is too early (urgent) it will be more costly because there is a need of greater processing speed and free resources to process particular request with urgency). There is individual price is fixed (artificially) for different resources because all the resources are working in coordination manner to fulfill the demand of user (demand of user is changing dynamically).

Experiment setup using 3 servers in which further virtual nodes (12 = 6 (Server 1) +4 (Server 2) +2 (Server 3)) are created. Every virtual node has different number for Execution Components (ECs) to process user request and every EC has their own cost (C$/EC time unit (Sec)). Table 1 shows the characteristics of the resources used and their Execution Component (EC) access cost per time unit in Cloud dollars (C$) and access cost in C$ is manually assigned for experimental purposes. The access cost of an EC in C$/time unit does not necessarily reflect the cost of execution when ECs have different capabilities. The execution agent needs to translate the access cost into the C$ for each resource. Such translation helps in identifying the relative cost of resources for executing user requests on them. Due to limited number of resources, cost increases with increase in user requests. Cost is varying in two different cases: i) relaxed deadline and ii) tight deadline. In both cases, when the deadline is low (e.g. 200 secs), the number of user requests processed increases as the budget value increases. When a higher budget is available, the execution agent uses expensive resources to process more user requests within the deadline. Alternatively, when scheduling with a low budget, the number of user requests processed increases as the deadline is relaxed. Different number of experiments has been performed by comparing AaaS (QoS-aware Autonomic) as discussed in *Section 4* with non-autonomic resource management technique (non-autonomic) in which no autonomic scheduling mechanism is considered while allocating resources to process the user requests.

5.1. Datasets

Datasets used in this research work are downloaded from the Open Government Data Platform India (data.gov.in, 2015), India Agriculture and Climate Data Set (Sanghi et al.), and regional land and climate modelling in China (Sanghi et al.) can be in the order of 1000000 records, with size of 3.5 GB. The data is coming in large data variety and volume from both users in the form of images like damaged crop images due to weather, insects etc. and devices through Internet of Things (IoT) sensors and satellites (GPS systems) that send weather related images. As a result of regular capturing and collection of datasets, they grow with the velocity of 80.72 KB/minute or more (Sanghi et al.). Five different tables used to process the different types of data as described in Table 8 to Table 12.

Table 7. Configuration Details of Cloud Environment

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

Table 8. Crop Information

CropId	Crop Name	Crop Type	Soil Texture	Min Land	Growing Period	Seed Type	Price	Quantity
C1	Rice	Kharif	Slity Clay	5 Acre	3 Months	Wet	1200 Rs./ Kg	2 Kg/Acre
C2	Maize	Rabi	Slity Loam Clay	4 Acre	4 Months	Dry	1600 Rs./ Kg	1 Kg/Acre
C3	Wheat	Zaid	Loam Clay	3 Acre	3 Months	Wet	1000 Rs./ Kg	2 Kg/Acre
C4	Sugarcane	Cash	Slity	4 Acre	6 Months	Dry	800 Rs./Kg	6 Kg/Acre

Table 9. Weather information

Crop Name	Temperature	Season	Pressure (CFM)	Wind Speed	Rainfall	Location
Rice	15-18 °C	Winter	0.75 to 1.5	16 Km/h	300–650 mm	Ambala
Maize	17-22 °C	Summer	0.05 to 0.5	12 Km/h	100–150 mm	Amritsar
Wheat	25-30 °C	Rainy	1.5 to 5.2	17.3 Km/h	200–250 mm	Ganga Nagar
Sugarcane	35-40 °C	Summer	1 to 10	8 Km/h	400–600 mm	Pathankot

Table 10. Soil information

Soil Texture	Bulk Density	Inorganic Material	Organic Material	Water	Air	Color	Structure	Infiltration
Slity Clay	2.60 to 2.75 grams per cm^3	Sand and clay	Plant and animal residues	25%	28%	Brown	Plate-like	15 mm/hour
Slity Loam Clay	2.7 to 2.75 grams per cm^3	Sand and Slit	Animal residues	22%	18%	Red	Prism-like	10 mm/hour
Loam Clay	2.60 to 2.75 grams per cm^3	Clay and Slit	Plant residues	37%	21%	Brown	Block like	18 mm/hour
Slity	2.60 to 2.75 grams per cm^3	Sand, Slit and Clay	Plant and animal residues	31%	29%	Black	Sphere like	22 mm/hour

Table 11. Pest information

Crop Type	Crop Disease	Effect	Treatment	Pesticide Name	Solubility in Water	Price	Outcome
Kharif	Bacterial brown spot	Degrade soil fertility	Reduce Irrigation	Carbonate	Yes	Rs. 1500/L	Improve Productivity
Rabi	Zonate eye spot	Degrade productivity	Distribute Soil	Organophosphate	No	Rs. 2200/L	Improve soil fertilization
Zaid	Dwarf bunt	Increase risk of other disease	Spray irrigation	Parathyroid	Yes	Rs. 2300/L	Reduce risk of other diseases
Cash	Ergot	Degrade productivity	Drip Irrigation	Parathyroid	Yes	Rs. 1800/L	Reduce productivity

Table 12. Fertilizer information

Crop Type	Fertilizer Name	Nutrient Composition	Price
Kharif	Urea	Nitrogen in form of urea (amide) (N)	7000 Rs./10 Kg
Rabi	Ammonium-Nitrate	Ammoniacal Nitrogen, Nitrogen Nitrate and Urea Nitrogen	9100 Rs./10 Kg
Zaid	Ammonium-Sulphate	Ammoniacal nitrogen and Sulpher	6200 Rs./10 Kg
Cash	Urea-Ammonium	Ammoniacal nitrogen and Neutral ammonium citrate Soluble phosphate	13200 Rs./10 Kg

5.2. Performance Metrics

The following metrics are used to calculate the execution cost, execution time, resource utilization, latency, detection rate and scalability for processing user requests as taken from previous work (Singh and Chana, 2015; Singh et al., 2015; Singh and Chana, 2016):

Execution Time is a ratio of difference of request finish time $\left(WF_i\right)$ and request start time $\left(WStart_i\right)$ to number of requests. Following formula is used to calculate Execution Time (ET) (Equation 3):

$$ET_i = \sum_{i=1}^{n}\left(\frac{WF_i - WStart_i}{n}\right) \tag{3}$$

where n is the number of requests to be executed.

Execution Cost is defined as the total amount of cost spent per one hour for the execution of request and measured in Cloud Dollars (C$). Following formula is used to calculate execution cost (C) (Equation 4):

$$C = ET_i \times Price \tag{4}$$

Latency is a defined as a difference of time of input cloud workload and time of output produced with respect to that workload. Following formula is used to calculate Latency (Equation 5):

$$Latency_i = \sum_{i=1}^{n}\left(time\,of\,output\,produced\,after\,execution - time\,of\,input\,of\,cloud\,workload\right) \tag{5}$$

where *n* is number of workloads.

Resource Utilization is defined as a ratio of actual time spent by resource to execute workload to total uptime of resource for single resource. Following formula is used to calculate resource utilization (Equation 6):

$$ResourceUtilization_i = \sum_{i=1}^{n}\left(\frac{actual\,time\,spent\,by\,resource\,to\,execute\,workload}{total\,uptime\,of\,resource}\right) \tag{6}$$

where *n* is number of workloads.

Security is measured in terms of detection rate. Experiment has been conducted with different type of attacks (DoS, R2L, U2R and Probing) and different tools used to launch different attacks are metasploit framework for DoS, Hydra for R2L, NetCat for L2R and NMAP for probing. *Detection Rate* is the ratio of total number of true positives to the total number of intrusions (Sorensen et al., 2010):

$$Detection\,Rate = \frac{Total\,Number\,of\,True\,Positives}{Total\,Number\,of\,Intrusions} \tag{7}$$

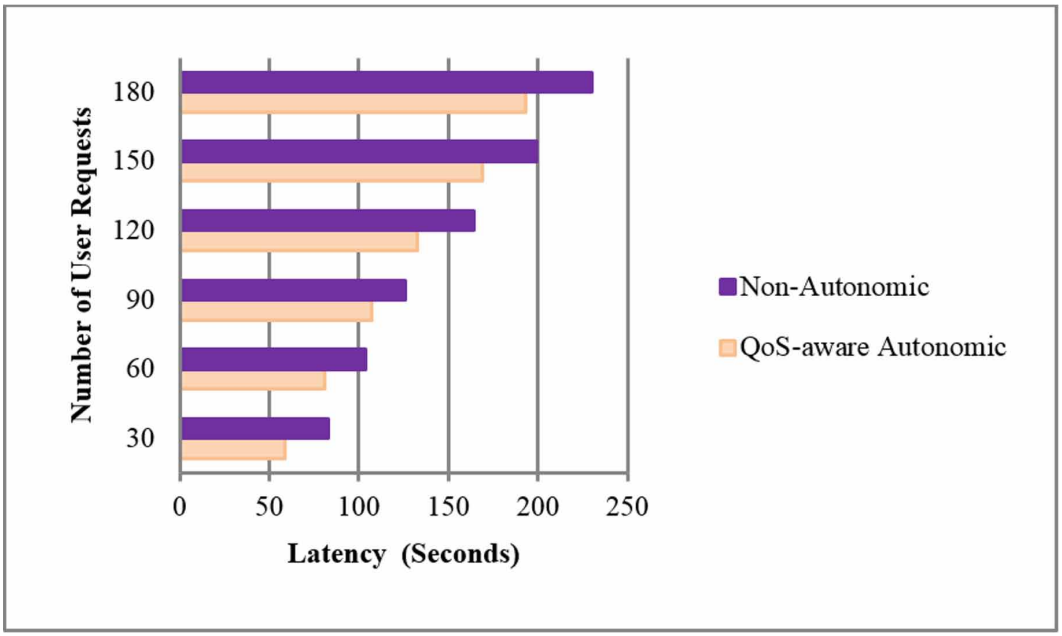
Scalability is measured in terms of throughput. It is the ratio of total number of workloads to the total amount of time required to execute the workloads. Following formula is used to calculate throughput (Equation 8):

$$Throughput = \frac{\text{Total Number of Workloads}\left(W_n\right)}{Total\,amount\,of\,time\,required\,to\,execute\,the\,workloads\left(W_n\right)} \tag{8}$$

5.3. Experimental Results -- Based on Modelling and Simulation using CloudSim

Experiment has been conducted with 180 user requests for verification of execution cost, execution time, resource utilization, latency, detection rate and scalability. With increasing the number of user requests, the value of latency is increasing. The value of latency in QoS-aware autonomic system is lesser as compared to non-autonomic based resource scheduling at different number of user requests as shown in Figure 6. The maximum value of latency is 193 seconds and minimum value of latency is 59 seconds in QoS-aware autonomic resource management technique. Average latency in QoS-aware autonomic is 15.22% lesser than non-autonomic resource management technique. The value of average cost for both QoS-aware cloud based autonomic resource management technique and

Figure 6. Effect of change in number of user requests on latency



non-autonomic resource management is calculated with different number of user requests as shown in Figure 7. Average cost is increasing with increase in number of user requests. At 180 user requests, average cost in QoS-aware autonomic is 25.36% lesser than non-autonomic resource management technique. QoS-aware autonomic performs excellent with different number of user requests. Execution cost in QoS-aware autonomic is 27.65% lesser than non-autonomic resource management technique.

As shown in Figure 8, the execution time is increasing with increase in number of user requests. At 90 user requests, execution time in QoS-aware autonomic resource management technique is 24.66% lesser than non-autonomic resource management technique. After 120 user requests, execution time increases abruptly in non-autonomic resource management technique but QoS-aware autonomic performs better than non-autonomic technique. Average execution time in QoS-aware autonomic is 18.960% lesser than non-autonomic resource management technique. With increasing the number of user requests, the percentage of resource utilization is increasing. The percentage of resource utilization in QoS-aware autonomic resource management technique is more as compared to non-autonomic resource management (non-autonomic) at different number of user requests as shown in Figure 9. The maximum percentage of resource utilization is 94.66% at 180 user requests in QoS-aware autonomic but QoS-aware autonomic performs better than non-autonomic technique. Average resource utilization in QoS-aware autonomic is 31.96% more than non-autonomic resource management technique.

Scalability is measured in terms of throughput. Number of software, network and hardware faults (fault percentage) has been injected to verify the throughput of the proposed system with 100 user requests. Figure 10 shows the comparison of throughput of both QoS-aware autonomic resource management approach and non-QoS based resource management technique (non-autonomic) at 100 user requests and it is clearly shown that QoS-aware autonomic performs better than non-autonomic. In this experiment, it has been found the maximum value of throughput at fault percentage 45% i.e. QoS-aware autonomic has 26% more throughput than non-autonomic. Detection rate increases with respect to time and it considers the number of blocked and detected attacks. For new attack or intrusion detection, database is updated with new signatures and new polices and rules are generated to avoid

Figure 7. Effect of change in number of user requests on execution cost

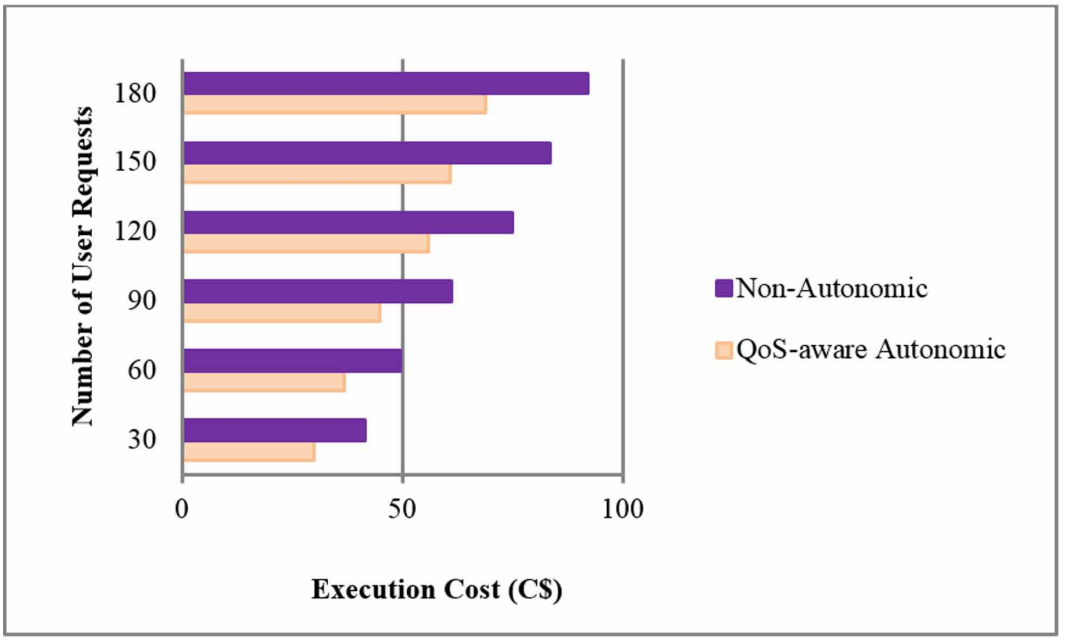


Figure 8. Effect of execution time with change in number of user requests

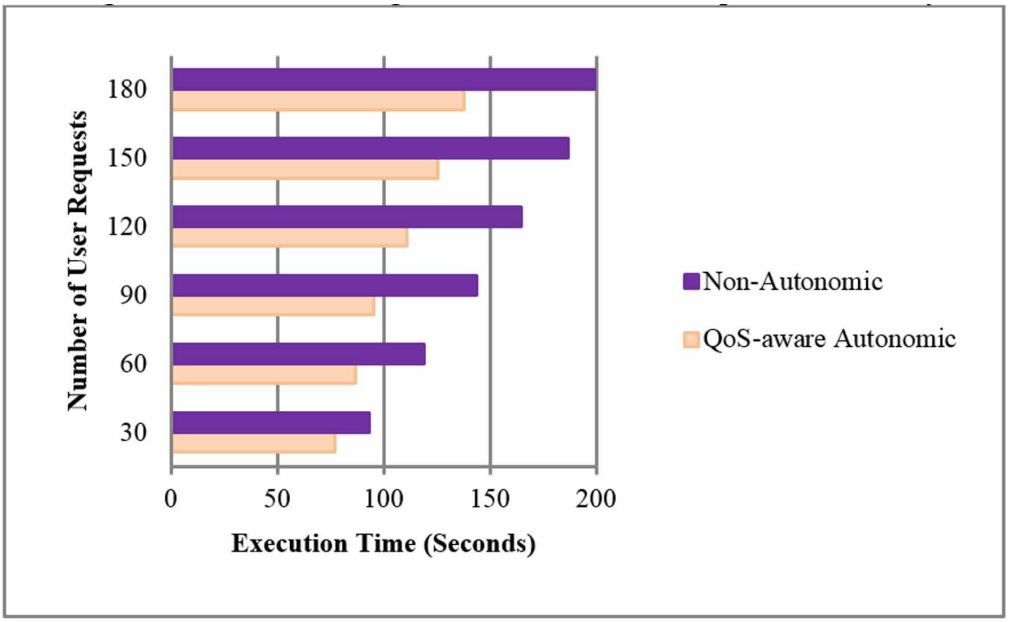
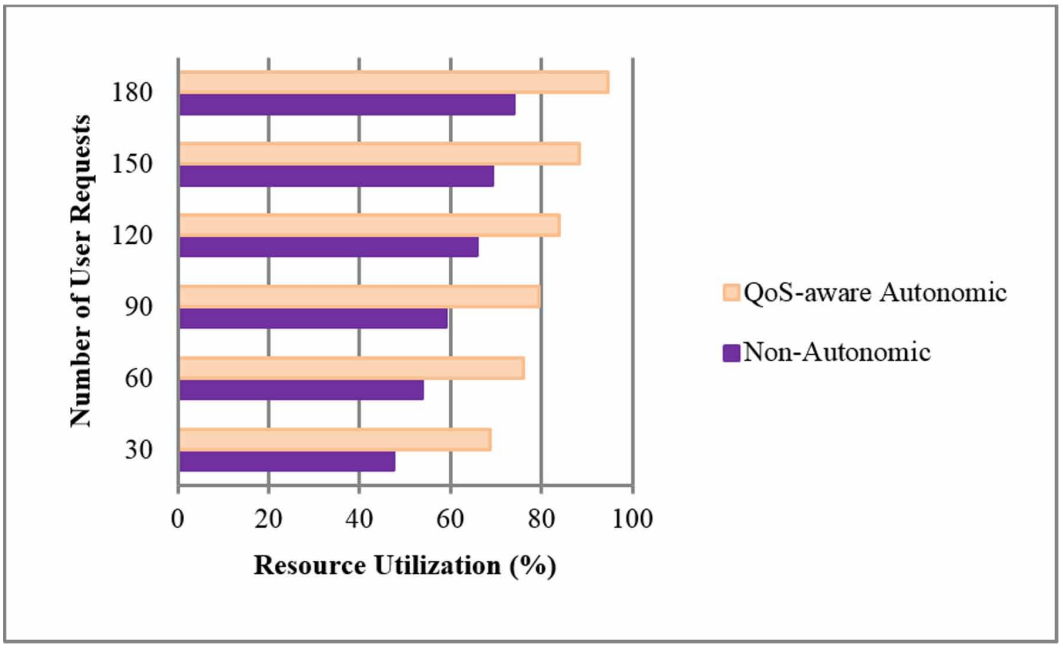


Figure 9. Effect of change in number of user requests on resource utilization



same attack. Experiment has been conducted for known attacks; it is clearly shown in Figure 11 that QoS-aware autonomic performs better than snort anomaly detector (non-autonomic). Further signatures of some known attacks have been removed from database to verify the working of proposed system.

Table 13 describes the comparison of execution time used to process different number of workloads (90 and 180) on cloud environment for proposed system with different number of

Figure 10. Throughput [100 user requests] vs. Fault percentage (%)

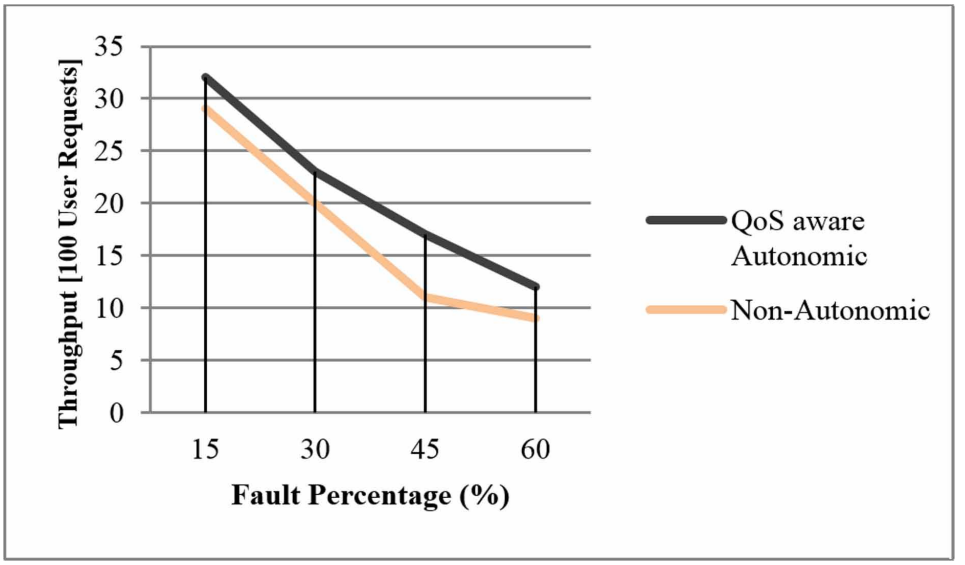
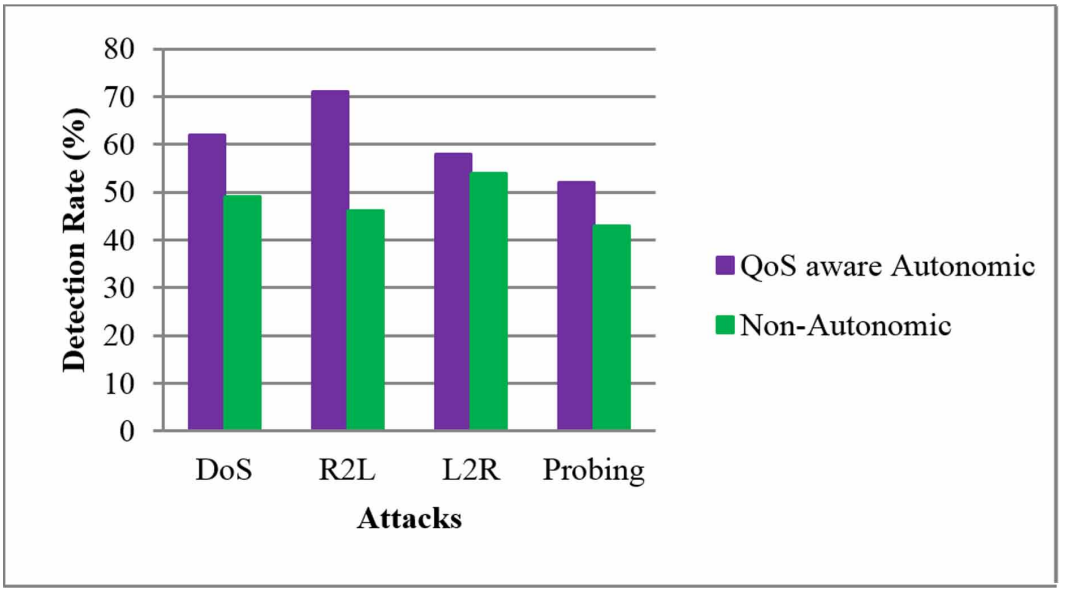


Figure 11. Detection rate vs. Attacks



Virtual Machines (VMs). The number of VMs used to execute the workloads was incremented gradually showing how the total execution time was reduced when more VMs were added to the cloud. With one virtual node running on Server R1, execution of 45 workloads finished in 436.12 seconds. With 12 virtual nodes (6 running on R1, 4 running on R2 and 2 running on R3), the application took 276.16 seconds. It is noted that the execution time is reduced with adding additional virtual nodes.

Table 13. Total execution time of a bulk of cloud workloads distributed in three servers

Number of Workloads	Virtual Nodes			Total Workers	Execution Time (Seconds)
	R1	R2	R3		
45	1	0	0	1	436.12
45	1	1	0	2	428.69
45	2	1	0	3	418.97
45	2	2	0	4	407.55
45	3	2	0	5	398.17
45	4	2	0	6	380.30
45	4	2	1	7	361.66
45	4	3	1	8	345.18
45	5	3	1	9	331.21
45	5	3	2	10	315.03
45	5	4	2	11	299.97
45	6	4	2	12	276.16
90	1	0	0	1	1803.11
90	1	1	0	2	1771.18
90	2	1	0	3	1759.66
90	2	2	0	4	1736.15
90	3	2	0	5	1691.77
90	4	2	0	6	1668.96
90	4	2	1	7	1636.11
90	4	3	1	8	1625.19
90	5	3	1	9	1578.21
90	5	3	2	10	1551.68
90	5	4	2	11	1529.11
90	6	4	2	12	1503.11

5.4. Statistical Analysis

Statistical significance of the results has been analyzed by Coefficient of Variation $\left(Coff.\ of Var.\right)$, a statistical method. $Coff.\ of Var.$ is statistical measure of the distribution of data about the mean value. $\mathrm{Coff.\ of\ Var.}$ is used to compare to different means and furthermore offer an overall analysis of performance of the technique used for creating the statistics. It states the deviation of the data as a proportion of its average value, and is calculated as follows (Equation 9):

$$Coff.\ of Var. = \frac{SD}{\mathrm{M}} \times 100 \tag{9}$$

where SD is a standard deviation and M is mean. $Coff.\ of Var.$ of execution time and have been studied of QoS-aware autonomic resource management technique and non-autonomic resource

management technique as shown in Figure 12 and Figure 13. Range of $Coff.\ of Var.$ (0.25% - 1.69%) for execution time and (0.37% - 1.96%) for cost approves the stability of QoS-aware autonomic resource management technique as shown in Figure 12 and Figure 13. Small value of $Coff.\ of Var.$ signifies QoS-aware autonomic resource management technique is more efficient in resource scheduling in the situations where the number of user requests has changed. Value of $Coff.\ of Var.$ decreases as the number of user requests is increasing.

6. CONCLUSION AND FUTURE DIRECTIONS

Cloud-based autonomic information system (AaaS) for agriculture service has been presented, which manages the various types of agriculture-related data based on different domains through different user preconfigured devices. K-NN (*k*-Nearest Neighbor) classification mechanism is used to classify the agriculture data. Further, classified data is interpreted and users can easily diagnose the agriculture status automatically through AaaS. In addition, AaaS uses two resource scheduling polices (time and cost) for efficient resource allocation at infrastructure level after identification of QoS requirements of user request. The performance of proposed system has been evaluated in cloud environment and

Figure 12. CoV for execution time with each scheduling algorithm

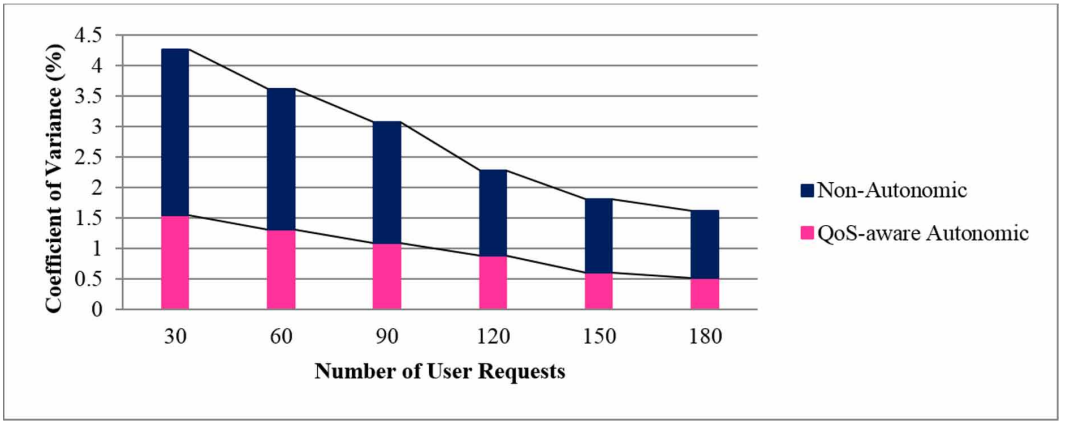
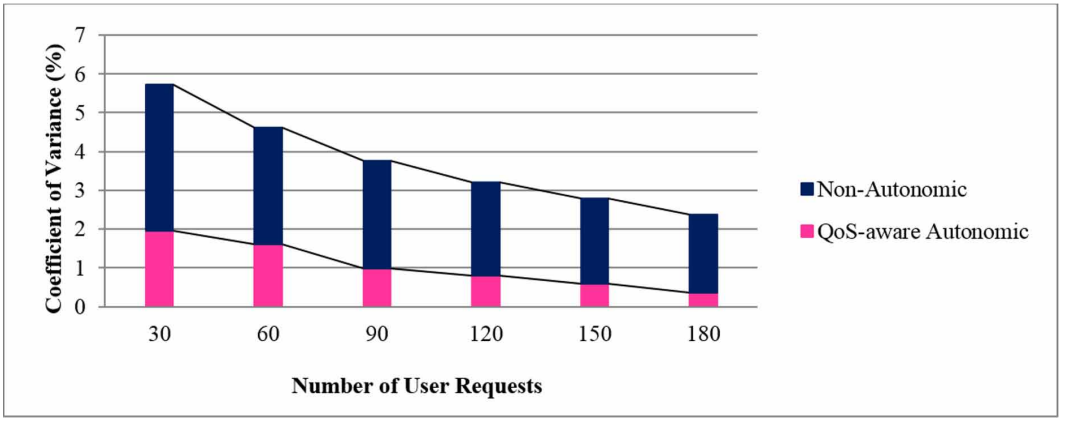


Figure 13. CoV for execution cost with each scheduling algorithm



experimental results show that the proposed system performs better in terms of execution time, cost, resource utilization, latency, scalability and security. In future, the proposed technique can be extended by incorporating other QoS parameters like network bandwidth, availability, customer satisfaction, computing capacity etc. Proposed technique can be extended by developing pluggable scheduler, in which resource scheduling can be changed easily based on the requirements.

ACKNOWLEDGMENT

One of the authors, Dr. Sukhpal Singh Gill [Post Doctorate Fellow], gratefully acknowledges the CLOUDS Lab, School of Computing and Information Systems, The University of Melbourne, Australia, for awarding him the Fellowship to carry out this research work.

REFERENCES

Agriculture Data, Government of India. (n. d.). Retrieved from https://data.gov.in/catalogs/sector/Agriculture-9212

Elsheikh, R., Mohamed Shariff, A. R. B., Amiri, F., Ahmad, N. B., Balasundram, S. K., & Soom, M. A. M. (2013). Agriculture Land Suitability Evaluator (ALSE): A decision and planning support tool for tropical and subtropical crops. *Computers and Electronics in Agriculture*, *93*, 98–110. doi:10.1016/j.compag.2013.02.003

Hu, Y., Quan, Z., & Yao, Y. (2004). Web-based Agricultural Support Systems. *Proceeding of the Workshop on Web-based Support Systems* (pp. 75-80).

India Agriculture And Climate Data Set. (n. d.). Retrieved from https://ipl.econ.duke.edu/dthomas/dev_data/datafiles/india_agric_climate.htm

Jeong, S., Jeong, H., Kim, H., & Yoe, H. (2013). Cloud Computing based Livestock Monitoring and Disease Forecasting System. *International Journal of Smart Home*, *7*(6), 313–320. doi:10.14257/ijsh.2013.7.6.30

Narayana Reddy, M., & Rao, N. H. (1995). *GIS Based Decision Support Systems in Agriculture*. National Academy of Agricultural Research Management Rajendranagar.

Nikkilä, R., Seilonen, I., & Koskinen, K. (2010). Software architecture for farm management information systems in precision agriculture. *Computers and Electronics in Agriculture*, *70*(2), 328–336. doi:10.1016/j.compag.2009.08.013

Prasad, S., Peddoju, S. K., & Ghosh, D. (2013). AgroMobile: A Cloud-Based Framework for Agriculturists on Mobile Platform. *International Journal of Advanced Science and Technology*, *59*, 41–52. doi:10.14257/ijast.2013.59.04

Ruixue, Z. (2002). Study on Web-based Agricultural Information System Development Method. *Proceedings of the Third Asian Conference for Information Technology in Agriculture*, China (pp. 601-605).

Shangguan, W., Dai, Y., Liu, B., Ye, A., & Yuan, H. (2012, February 29). A soil particle-size distribution dataset for regional land and climate modelling in China. *Geoderma*, *171*, 85–91. doi:10.1016/j.geoderma.2011.01.013

Singh, S., & Chana, I. (2015). QoS-aware Autonomic Resource Management in Cloud Computing: A Systematic Review. *ACM Computing Surveys*, *48*(3), 1–46. doi:10.1145/2843889

Singh, S., & Chana, I. (2015). QRSF: QoS-aware resource scheduling framework in cloud computing. *The Journal of Supercomputing*, *71*(1), 241–292. doi:10.1007/s11227-014-1295-6

Singh, S., & Chana, I. (2015). Q-aware: Quality of service based cloud resource provisioning. *Computers & Electrical Engineering*, *47*, 138–160. doi:10.1016/j.compeleceng.2015.02.003

Singh, S., & Chana, I. (2016). EARTH: Energy-aware Autonomic Resource Scheduling in Cloud Computing. *Journal of Intelligent and Fuzzy Systems*, *30*(3), 1581–1600. doi:10.3233/IFS-151866

Sørensen, C. G., Fountas, S., Nash, E., Pesonen, L., Bochtis, D., Pedersen, S. M., & Blackmore, S. B. et al. (2010). Conceptual model of a future farm management information system. *Computers and Electronics in Agriculture*, *72*(1), 37–47. doi:10.1016/j.compag.2010.02.003

Sørensen, C. G., Pesonen, L., Bochtis, D. D., Vougioukas, S. G., & Suomi, P. (2011). Functional requirements for a future farm management information system. *Computers and Electronics in Agriculture*, *76*(2), 266–276. doi:10.1016/j.compag.2011.02.005

Sukhpal Singh Gill joined Computer Science and Engineering Department of Thapar University, Patiala, India, in 2016 as a Faculty. Presently, Dr. Gill is working as Post Doctorate Fellow at CLOUDS Lab, School of Computing and Information Systems, The University of Melbourne, Australia. Dr. Gill obtained the Degree of Master of Engineering in Software Engineering from Thapar University, as well as a Doctoral Degree specialization in "Autonomic Cloud Computing" from Thapar University. Dr. Gill received the Gold Medal in Master of Engineering in Software Engineering. Dr. Gill is a DST Inspire Fellow [2013-2016] and worked as a SRF-Professional on DST Project, Government of India. He has done certifications in Cloud Computing Fundamentals, including Introduction to Cloud Computing and Aneka Platform (US Patented) by ManjraSoft Pty Ltd, Australia and Certification of Rational Software Architect (RSA) by IBM India. His research interests include Software Engineering, Cloud Computing, Internet of Things and Fog Computing. He has more than 40 research publications in reputed journals and conferences.

Inderveer Chana joined Computer Science and Engineering Department of Thapar University, Patiala, India, in 1997 as Lecturer and is presently serving as Professor in the department. She is Ph.D. in Computer Science with specialization in Grid Computing, M.E. in Software Engineering from Thapar University and B.E. in Computer Science and Engineering. Her research interests include Grid and Cloud computing and other areas of interest are Software Engineering and Software Project Management. She has more than 100 research publications in reputed Journals and Conferences. Under her supervision, more than 40 ME thesis and seven Ph.D thesis have been awarded and five Ph.D. thesis are on-going. She is also working on various research projects funded by Government of India.

Rajkumar Buyya is a Fellow of IEEE, Professor of Computer Science and Software Engineering and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercialising its innovations in Cloud Computing. He has authored over 500 publications and four text books. He is one of the highly cited authors in computer science and software engineering worldwide (h-index 110+, 60000+ citations). He has served as the founding Editor-in-Chief (EiC) of IEEE Transactions on Cloud Computing and now serving as Co-EiC of Journal of Software: Practice and Experience.