IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-9, Sept 2019 E-ISSN: 2347-2693

Community Cloud Model for Infrastructure –As-A-Service in Learning through Content Sharing

Vincent Mbandu Ochango[1*], Waweru Mwangi[2], George Okeyo[3]

[1,2,3]School of Computing and Information Technology, Jomo Kenyatta University of Agriculture and Technology
Nairobi, Kenya

[*]*Corresponding Author: ochangovincent@gmail.com, Tel.: +254-07-02-86-49-74*

DOI: https://doi.org/10.26438/ijcse/v7i9.17 | Available online at: www.ijcseonline.org

Accepted: 12/Sept/2019, Published: 30/Sept/2019

Abstract—Cloud computing is growing rapidly, with its server farms evolving at a remarkable rate. This paper tries to elaborate on a community cloud model that tries to use web application to identify maize disease through image matching. The model tries to match uploaded images with the images stored in the file system and if a match is found, the matching image is displayed together with its associated disease. The images were matched based on time, location and images collected from secondary sources to see if the disease can still be identified. All the images matched were successful and the appropriate diseases associated with the images were identified. The load balancing technology was also incorporated into the model mainly for the accessibility of the web application and to avoid overloading of one server. The image matching results were collected and tabulated as shown in chapter four and the results clearly indicated that the community cloud model really helped users to identify possible disease through the web application hosted on the community cloud model servers.

***Keywords*—**Cloud Computing, Infrastructure-as-a-Service, Web server, Image Matching, Load Balancing

I. INTRODUCTION

Well, known Web applications cannot depend on one server since when the load exceeds one server gets overloaded hence one server can process requests which leads to users experiencing delays [1]. The problem to be solved in this paper was that one to be able to identify a disease after image matching through the web application hosted on community cloud model servers. There is also high load while users try to access the web application which leads to overloaded servers that cant process requests on time which in turn leads to delays hence affecting the clients. Load balancing technology was incorporated into the model which distributed requests among upstream servers hence reducing server overloading. Distributed Web server models that straightforwardly plan customer demands offer an approach to meet powerful adaptability and accessibility necessities hence this avoids overloading the servers hence reducing delays which leads to high throughput among servers [2].
The aim of this paper is to show and examine how the community cloud model can be used to identify disease after image matching by using a web-based application hosted on servers.

A community cloud model aids counterbalance normal difficulties crosswise over organizations, such as cost, innovation multifaceted nature, and spending prerequisites, security concerns and an absence of division explicit administrations from specialist suppliers [3].

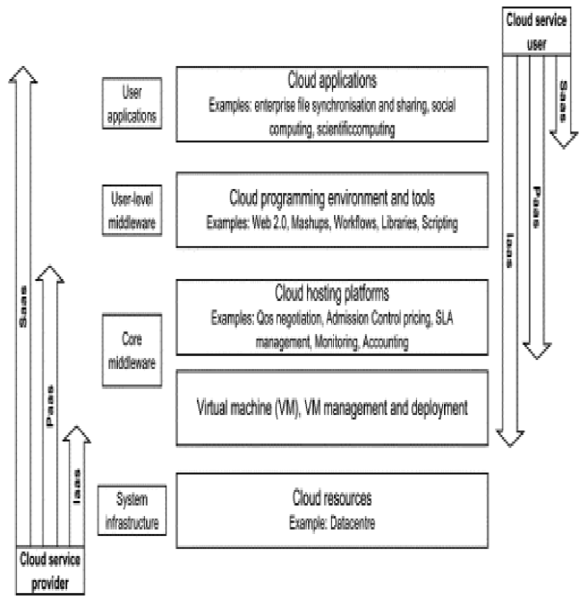
Rest of the paper is organized as follows, Section I contains the introduction of community cloud model, Section II contain the related work of cloud computing infrastructure and community cloud, Section III contain the methodology with proposed community cloud model, Section IV describes experimental results and discussion, Section V concludes research work with future directions).

II. RELATED WORK

2.1 Cloud Computing Infrastructure

Several cloud stacks are part of the cloud computing infrastructure [4]. To form a datacenter you require hundreds to thousands of nodes which is the part of cloud resources or the lowest stack. In order to build the infrastructure that is distributed, with Hosting Platform of cloud supports the main responsibilities of management infrastructure like metering usage, accounting and billing the virtualization technology deployed in core middleware.

The core middleware and the underlying system infrastructure is where the Infrastructure as a Service is formed from. The development platform is a cloud service offered at the user-level middleware which is referred to as PaaS. The SaaS sends cloud applications to CSU and it is usually at the top stack which entails user applications [5].


Cloud service user
Saas
Paas
Iaas
User applications
Cloud applications
Examples: enterprise file synchronisation and sharing, social computing, scientificcomputing
User-level middleware
Cloud programming environment and tools
Examples: Web 2.0, Mashups, Workflows, Libraries, Scripting
Core middleware
Cloud hosting platforms
Examples: Qos negotiation, Admission Control pricing, SLA management, Monitoring, Accounting
Virtual machine (VM), VM management and deployment
System infrastructure
Cloud resources
Example: Datacentre
Cloud service provider

The tasks and obligation of resources of the cloud are separated amongst Cloud Service Provider and Cloud Service User which makes the difference between cloud and traditional in-house infrastructure [6].

In the Figure above, the architecture of cloud heap from Buyya et al. (2015) was adopted with the possibility of the controller from Jansen and Grance (2013) to expound on the scope of control amongst the CSP and CSU, for all heap of the cloud structural design. The arrows represent the range of scope and control over the stack of resources. When the stack gets lower the Cloud Service User should have extra access to cloud resources. Cloud Service Provider will take more control with Software as a Service however minimum control with Infrastructure as a Service then contrariwise Cloud Service User should take less control over the resources with Software as a Service then extra control with Infrastructure as a Service. The CSU or CSP capability to create and handle mechanisms of security is determined by control over resources. The CSP is responsible for hypervisor vulnerability management while the CSU is responsible for secure coding, and encryption under PaaS level. During planning cloud incident handling strategies, the security control shared responsibility handling and controllable needs to be taken into consideration [7].

Among data centers of the same CSP, a workload can migrate from one server to another server. At this time it is not feasible moving a load to a CSP which is different. For all targeted clouds, application binaries must be created and data replicated in order to use multiple clouds. This is an unfeasible in practice thus, a costly proposition. Information about data formats, software procedures, tools executing these procedures and software stack internal specifications, Cloud Service Providers are usually difficult to share. Such data gives CSP an advantage over its competitors [8]. Cloud interoperability poses a fair number of challenges due to the current limitations of computing and communication technologies. The growth of an interacting cloud, a global company permitting Cloud Service Providers to distribute workload is quite difficult while why a difficult system such as the interconnection of computers is very effective? The architecture of the Internet which is a global computer network is built on two acceptable concepts:

- The computer at the margin of the global computer network, need to have one or more Internet Protocol address so that each computer in a network to be identified easily.;
- Every computer must send packets by similar IP, thus packets transmitted must arrive its endpoint.

The interconnection of computers is to send packets irrespective of their origin, composition, vocal sound, pictures, packets received by a measuring device, or credible kind of data. The packets are packed in small blocks, large, medium, or desirable size of blocks and also should be packaged again at whatever time the necessity arises in order to create the circumstances at ease. The packets are delivered from the origin to the endpoint within a short amount of time in case the software requires so [9].

2.2 Community Cloud

A community cloud is incorporating the variety of services of different clouds to handle the exact concerns of a community through distributed systems [10]. The community cloud infrastructure is used by the community that makes use of the cloud services offered by the cloud. The cloud of a certain community should exist within their premises or outside their premises and it should be controlled by the community itself or an external organization [11].

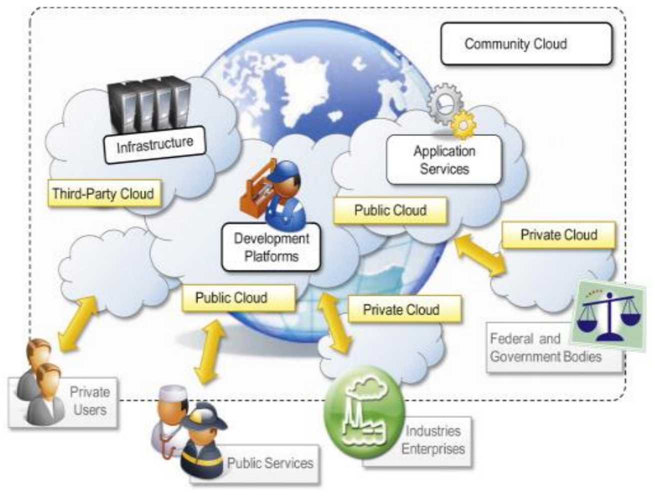


Figure 1. Usage Scenario of Community Clouds

Figure 1 shows the reference architecture together with the general view of the usage scenario of community clouds. All of the users focus on the same issues for their interaction with the cloud, the community cloud users fall into a well-identified community, sharing the same concerns or needs; they can be government bodies, industries, or even simple users [12]. The community cloud serves a multitude of users with different needs which is a different scenario than public clouds [13]. With community cloud, the services are generally delivered within the institution that owns the cloud which makes it also different from private clouds.

III. METHODOLOGY

The main motivation for creating the community cloud model was to enable users to identify the possible disease associated with the image they upload through the web application that is hosted on the community cloud servers. The community cloud servers have been incorporated with the load balancing technology that reduces the overloading of servers hence enhancing the accessibility of the web application hosted on the community cloud servers.

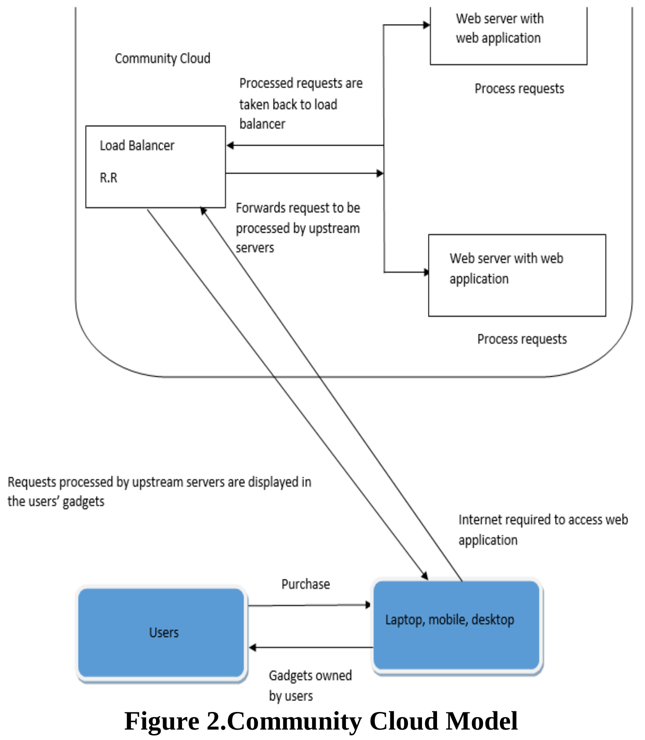

Community Cloud
Web server with web application
Process requests
Processed requests are taken back to load balancer
Load Balancer R.R
Forwards request to be processed by upstream servers
Web server with web application
Process requests
Requests processed by upstream servers are displayed in the users' gadgets
Internet required to access web application
Purchase
Users
Laptop, mobile, desktop
Gadgets owned by users

Figure 2.Community Cloud Model

Figure 2 above is a theoretical model for community cloud, to really make use of the model, the users must be existing. When the user has a gadget that is able to access the internet, he or she requests for web application through his gadget. The request first hits the webserver distributing requests and the server itself has been configured with the Round Robin (R.R) method of distributing requests. The web server will then forward the first load to the first listed upstream server which will process the requests and takes back to the webserver. If the second requests hit the webserver distributing requests which will, in turn, the forward second request to the second listed web server. When the requests will continue hitting the web server distributing requests, the webserver shall continue forwarding them in a circular fashion. The web server will then take back the processed requests to the users' gadget. The users will, in turn, be able to view the web application on their gadget. The users will be provided with an interface through the web application where they will be able to upload the image they have taken. And the image uploaded is matched with the one in the web application file system. If the match is found the disease associated with the image matched with is displayed to the user.

3.1 Round Robin Algorithm

The round-robin was used as the way of distributing load for the model. Community cloud model had three servers in place. The first server was configured with the round-robin load-balancing algorithm and the other two servers acted as the upstream servers. The round-robin load-balancing algorithm rotated connection requests among web servers in the order that requests were received. For a simplified example, the research work had a cluster of two servers: emaize2 and emaize3.

- The first request went to emaize2 server.
- The second request went to emaize3 server.

The load balancer continued passing requests to servers in a round-robin manner.
During configuration of the load balancer with the algorithm, the following was considered;

1. The servers to be included in the load balancing scheme were defined
2. The research work used the servers' private IPs for better performance and security.
3. The servers private IPs were found at the Up Cloud control panel Network section.
4. The research work ensured that the load balancer accepted all traffic to port 80 and passed the traffic to the upstream servers.
5. The research work also ensured that the upstream name and the proxy_pass matched and the upstream name used was emaize.

3.1.1 The load balancer tool used to collect web server metrics

Nginx Amplify Agent tool was installed in the load balancer (Load balancer is a web server) to provide system and web server metric collection. This enabled the research work to know the number of connection accepted against the time the connection was accepted and from the results, it is clear to know if the load balancer accepted all the requests and forwarded them to the backend servers hence if all requests were forwarded there were no requests that were dropped.

Since there was no request dropped it showed that there was a connection established between the client and the server hence maize disease identification using the emaize web application was successful. The tool also enabled research work to collect the HTTP 5xx errors which refer to an error code 500-511, which covers server errors. These errors appear when the web application's server is failing to fulfil requests, and therefore the web application cannot display the requested data. From the results discussed in chapter four, there were zero HTTP 5xx errors this indicated that users were able to access emaize web application thus they were able to identify the possible disease according to the image they uploaded on the emaize web application.

3.2 Image matching with the emaize web application.

The web application developed and hosted on the digital ocean which is an Infrastructure-as-a-Service was called emaize. The emaize web application tries to match the image uploaded by the users of the web application with the one in the file system and if the match is found then the disease associated with the image matched is identified. The emaize web application was hosted on two servers and the third server was configured as a load balancer to avoid the overloading of the servers. The file system used contained 16 images that were matched with the one the users uploaded on the emaize web application. The proposed model targeted any user who practices maize farming and the model was mainly for users to be able to identify the disease through image matching according to the image they upload on emaize web application hosted on community cloud servers. A total of 22 images were uploaded using the emaize web application to see if the disease can be identified after image matching and the results were discussed in chapter four.

3.2.1 How image matching was done with emaize web application.

Pseudocode

```
Begin
 For all images
        Convert them to bitmap
If bitmap images are valid then
        Resize them to 16×16 pixel
Else if images are equal then
        Convert them to grayscale
Else if images are grayscale then
        Loop through each pixel to compare
Else
        Images not grayscale
End if
End for
Calculate the percentage difference
Similarity=100%-Percentage Difference
Return the highest similarity
Return the name of the image with the highest similarity
End
```

3.3 Images Used During Experiment

The images used for the experiment were 16 images which were stored in a file system, 4 images were stored in folders named Corn Smut, Maize Chlorotic Mottle Virus, Maize dwarf mosaic, and Maize Lethal Necrosis respectively which leads to a total of 16 images since each folder had 4 images. The folders were named based on the disease the images are associated to.

The 16 images stored in the file system were obtained from both secondary and primary sources and were matched with images collected based on location, time and secondary sources.

First of all the research work started by matching images based on the location they were taken from, time and finally images collected from secondary sources.

3.3.1 Images collected based on Location

Image is taken from Ebunangwe, Kenya

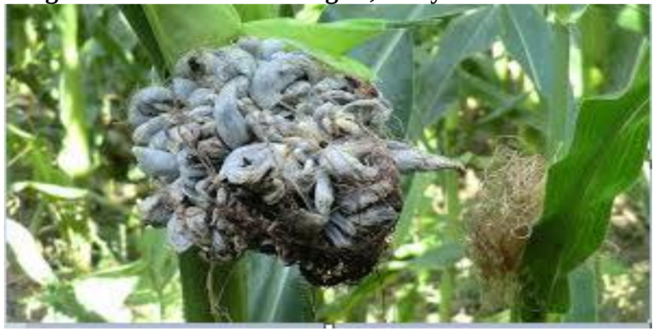


Image is taken from Ebusiloli, Kenya

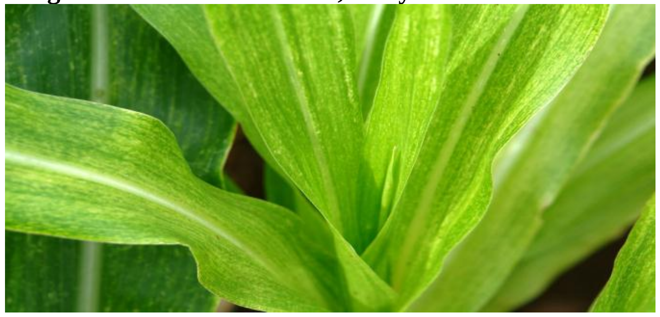
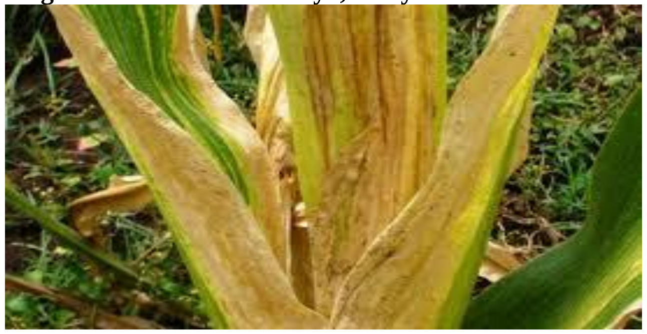


Image is taken from Emuhaya, Kenya



3.3.2 Images collected based on time

Morning Hours

Afternoon Hours

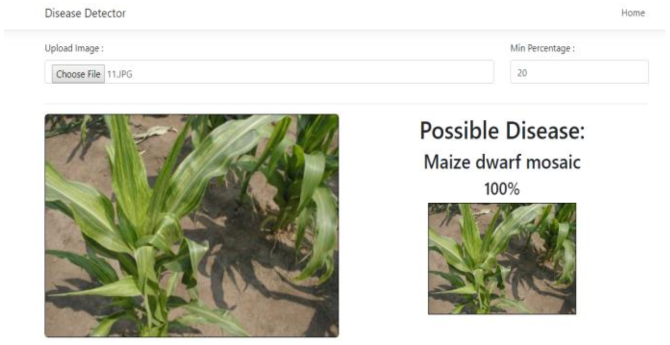
The above images together with the images collected from secondary sources were used in image matching experiment and the results were discussed in chapter four.

IV. RESULTS AND DISCUSSION

The main objective of this chapter was to explain the results of the model after being tested to see if it really does what it was supposed to do.

4.1 Image matching using the emaize application to identify maize disease.

The research work used emaize web application, where the emaize web application provides an interface where the farmers uploads the images and then the uploaded images are compared with the one in the file system then the appropriate percentage matching is returned with the appropriate disease the image could be associated with as shown in the diagram below.



Disease Detector Home

Upload Image : Choose File 11.JPG

Min Percentage : 20

Possible Disease:

Maize dwarf mosaic

100%

Figure 3.Maize Dwarf Mosaic disease identification from an image taken from Luanda, Kenya

The other images are taken from different places in Kenya to be matched and the percentage matching to be returned with the appropriate disease the image could be associated with is as shown in figure 4.

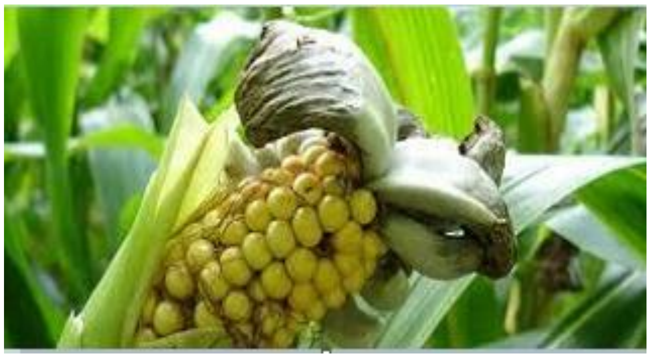
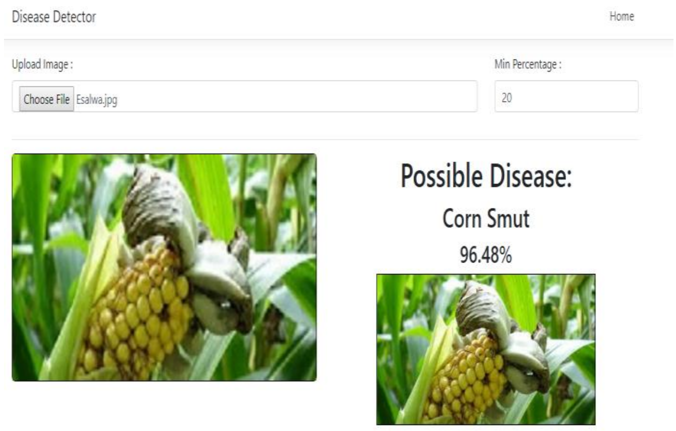


Figure 4.Image taken from Esalwa, Kenya



Disease Detector Home

Upload Image : Choose File Esalwa.jpg

Min Percentage : 20

Possible Disease:

Corn Smut

96.48%

Figure 5. Disease identified from figure 4 after image matching

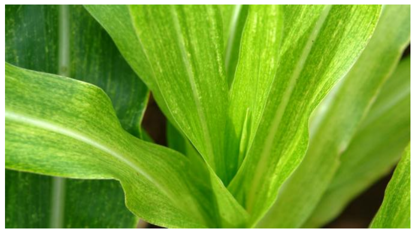
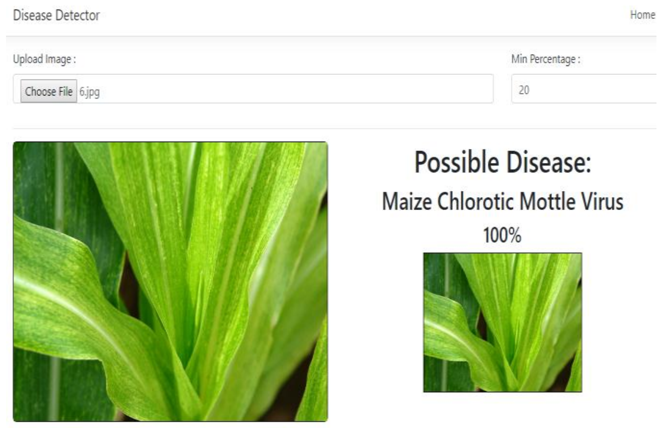


Figure 6.Image is taken from Ebusiloli, Kenya



Disease Detector Home

Upload Image : Choose File 6.jpg

Min Percentage : 20

Possible Disease:

Maize Chlorotic Mottle Virus

100%

Figure 7. Disease identified from figure 6 after image matching

Results of image matching based on time

The first image in the first category of image collected during morning hours was matched and the results were displayed as shown in figure 8.

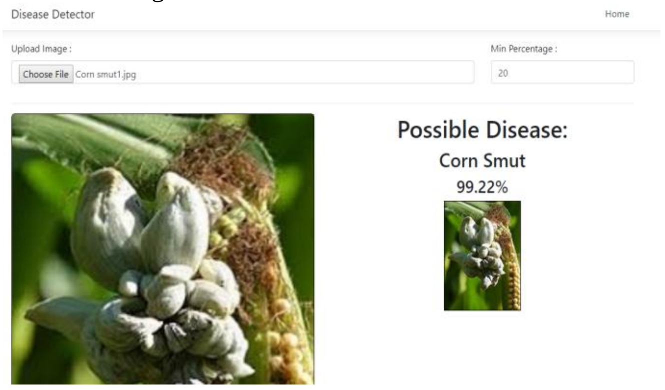


Figure 8.Corn smut disease

Figure 8 indicates that the first image taken under morning hours was associated with corn smut disease with a matching percentage of 99.22%

The image matching results gotten from emaize web application was tabulated as shown in the table below.

Table 1. Location image matching and possible disease identified

Image Source	Matching Percentage (%)	Possible Disease
Luanda, Kenya	100	Maize Dwarf Mosaic Virus
Esalwa, Kenya	96.48	Corn Smut
Ebusiloli, Kenya	100	Maize Chlorotic Mottle Virus
Mundika, Kenya	94.14	Maize Lethal Necrosis
Ebunangwe, Kenya	100	Corn Smut
Mwiyala,Kenya	97.66	Maize Chlorotic Mottle Virus

Table 2.Image matching based on images taken during morning hours and possible disease identified

Matching Percentage (%)	Possible Disease
99.22	Corn Smut
95.7	Maize Dwarf Mosaic

Table 3.Image Matching based on images taken during afternoon hours and possible disease identified

Matching Percentage (%)	Possible Disease
97.66	Corn Smut
94.92	Maize Dwarf Mosaic

Table 4.Image matching based on Images taken during evening hours and Possible Disease Identified

Matching Percentage (%)	Possible Disease
96.88	Corn Smut
93.75	Maize Dwarf Mosaic

Table 5. Image matching based on images taken from secondary sources and possible disease identified

Image Source	Matching Percentage (%)	Possible Disease
www.pri.org	97.66	Corn smut
en.wikipedia.org	99.61	Corn smut
www.flickr.com	99.22	Maize Chlorotic Mottle Virus
ohioline.osu.edu	98.05	Maize Dwarf Mosaic
b4fa.org	96.88	Maize Lethal Necrosis
core.ac.uk	98.83	Maize Chlorotic Mottle Virus
www.gardeningknowhow.com	98.44	Maize Dwarf Mosaic
www.talkafrica.co.ke	99.22	Maize Lethal Necrosis
www.hobbyfarms.com	97.27	Corn Smut

4.2 Nginx Amplify Agent used to collect server performance metrics.

The Amplify Agent is a Python application that provided a system and web server metric collection. Figure 9 shows HTTP 5xx errors which refer to an error code 500-511, which covers server errors. These errors appear when the web application's server is failing to fulfil requests, and therefore the web application cannot display the requested data. From figure 9 below it indicates that there were zero HTTP 5xx errors hence the users were able to identify the possible disease according to the image they uploaded on the emaize web application.

The request time per second from the figure below is 0.515s which means that the load balancer wasn't going through the event loop while processing the request. It was able to read the request from the client, read data from disk and send all data to backend servers without overflowing.

Figure 9 also indicates the health score of the emaize application. The health score of the emaize application is a calculated representation of how well the application is working based on its performance and resource utilization. According to the nginx amplify agent tool when the health score of the application is between 90 – 100 it indicates that it performs well. The health score for the emaize application according to figure 9 was 97.8% which is an indication that the end-user experience while interacting with the emaize web application was good.

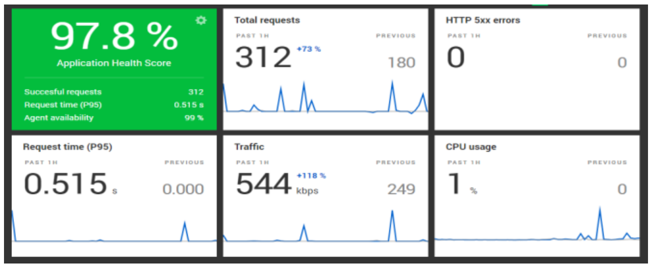


Figure 9. Overview representation of performance metrics

V. CONCLUSION AND FUTURE SCOPE

The fundamental target of the study was to create a community cloud model for farmers that identifies a possible disease from the image the farmer has uploaded to the emaize cloud-based application. The images uploaded by the farmers during testing of the model were images taken from different hours of the day and from a different location and from different secondary sources. The main aim of this was to see if the disease can still be identified despite images taken from a different time, location and different secondary sources. And from the experimental results and discussion, it indicates that the disease was identified from all the 22 images uploaded on the emaize web application. The community cloud model has used load balancing solution which has applied redundantly servers which help in better distribution of the communication traffic so that the emaize web application availability is conclusively settled. The round-robin load-balancing algorithm utilized on the community cloud model has been utilized to improve the accessibility and to lessen the overloading of the web servers. The emaize cloud-based application has made a great success and it provides quick and convenient information on the possible disease the image uploaded could be associated with. In future enhancements, the load balancing algorithm used can be fine-tuned further to consider different server processing capabilities.

ACKNOWLEDGMENT

I would like to thank my supervisors Prof.Waweru Mwangi and Dr.George Okeyo for their guidance and support in assisting me with the entire research. They have been with me through high and low moments I experienced during research. I am very grateful to them and may God bless you. Special acknowledgement to my uncle Benson Abwao, who supported me when I started this degree. Finally, to my mother Sarah Agiso Ochango for her support and encouragement throughout my studies.

REFERENCES

[1] Alhakami, H., Aldabbas, H., & Alwada, T. (2012). COMPARISON BETWEEN CLOUD AND GRID COMPUTING : REVIEW PAPER, 2(4), 1–21.
[2] Al Nuaimi K, Mohamed N, Al Nuaimi M, Al-Jaroodi J 2012 A survey of load balancing in cloud computing: challenges and algorithms *In Proceedings - IEEE 2nd Symposium on Network Cloud Computing and Applications, NCCA 2012* p. 137–42
[3] Ahmed, M., & Hossain, M. A. (2014). CLOUD COMPUTING AND SECURITY ISSUES IN THE Cloud. International Journal of Network Security & Its Applications, 6(1), 25–36. http://doi.org/10.5121/ijnsa.2014.6103
[4] Arokia, R., Rajan, P., & Shanmugapriyaa, S. (2013). Evolution of Cloud Storage as a Cloud Computing Infrastructure Service. IOSR Journal of Computer Engineering (IOSRJCE), 1(1), 38–45. Retrieved from http://arxiv.org/abs/1308.1303
[5] Badidi, E. (2013). A Framework for Software-as-a-Service Selection and Provisioning. International Journal of Computer Networks and Communications (IJCNC), 5(3), 12. http://doi.org/10.5121/ijcnc.2013.5314
[6] Basmadjian, R., Meer, H., Lent, R., & Giuliani, G. (2012). Cloud computing and its interest in saving energy: the use case of a private cloud. Journal of Cloud Computing: Advances, Systems, and Applications, 1(1), 5. http://doi.org/10.1186/2192-113X-1-5
[7] Computing, M., Jaiswal, P. R., & Rohankar, A. W. (2014). Infrastructure as a Service : Security Issues in Cloud Computing, 3(3), 707–711.
[8] Dave S, Maheta P 2014 Utilizing round robin concept for load balancing algorithm at virtual machine level in cloud environment *Int J Comput Appl* [Internet] 94(4) 23–9
[9] Desai T, Prajapati J 2013 A survey of various load balancing techniques and challenges in cloud computing *Int J Sci Technol Res* [Internet] 2(11) 158–61
[10] Distributed load balancing algorithms for cloud computing *24th IEEE Int Conf Adv Inf Netw Appl Work WAINA* [Internet] 551–6
[11] Gopinath P P G, Vasudevan S K 2015 An In-depth analysis and study of load balancing techniques in the cloud computing environment. *Procedia Comput Sci* [Internet] Elsevier Masson SAS 50 427–32
[12] Gulati, A., & Chopra, R. K. (2013). Dynamic round robin for load balancing in a cloud computing. *IJCSMC*, *2*(6), 274-278.
[13] Haryani N, Jagli D, Sangita O, Dhanamma J, Jagli1 M D, Solanki R, et al. 2014 Dynamic Method for Load Balancing in Cloud Computing *Int Conf Circuits, Syst Commun Inf Technol Appl* [Internet] 5(4) 336–40

Authors Profile

Vincent Mbandu is a passionate tech developer with experience leading in Information Technology, technology consultancy and Cloud Computing. He is undertaking Masters of Science in Information Technology from Jomo Kenyatta University of Agriculture and Technology. He holds a Bachelors of Science in Information Technology from JKUAT (2014).He is currently a Graduate Teaching Assistant at Gretsa University.

Waweru Mwangi is an associate professor of computing at Jomo Kenyatta University of Agriculture and Technology (JKUAT) in Kenya.

George Okeyo is a Lecturer in Computer Science at De Montfort University (DMU), Leicester, UK.