Mobile-Cloudlet Face Recognition: Two Different Approaches

Praseetha V. M.[1*], Ankit Bansal[2], Vadivel S.[1]

[1] BITS Pilani Dubai Campus, International Academic City, Duabi, United Arab Emirates.
[2] University of Michigan, Ann Arbor, USA.

* Corresponding author. Tel.: +971 529807160, +91 9447603338; email: praseethasunil@gmail.com
Manuscript submitted July 02, 2016; accepted January 9, 2017
doi: 10.17706/jcp.13.1.116-129

Abstract: The resource scarcity of mobile devices, pulls them back from executing the computation intensive computer vision applications like face recognition, augmented reality, video processing etc. As a solution cyber foraging, where mobile devices offload either computation or data or both to a high performance system within their network, can be applied. Cloudlet is a new emerging technique that scales up the computational ability of mobile devices to handle the high computation requirement. It allows mobile client to communicate with nearby available server and the minimal processed data can be transferred from mobile app to the server which will do the necessary processing and result can be send back to mobile in real time. This paper explains two novel approaches of successful implementation of principal component analysis (PCA) based face recognition using VM Based Cloudlet on mobile device.

Key words: Cloudlets, cloud computing, code offloading, cyber foraging, face recognition, mobile cloud computing, principal component analysis, VM synthesis.

1. Introduction

The most desirable features of mobile devices are small size, light weight, long battery life, comfort and less heat dissipation. Secondary concerns are system features like processor speed, storage capacity and memory size [1]. Whatever be the advancement in mobile technology, these devices are still having lot of limitations. Mobile cloud computing is considered as the solution for resource limitations of the mobile device in which the compute intensive tasks are offloaded to the cloud. These tasks are processed in the cloud with the help of resource rich devices in the cloud and the result is given back to the mobile device. Thus the limitations of the mobile device are alleviated by using various augmentation strategies.

Mobile computing is introduced to achieve the developing technology proposed by pervasive computing [2]. Many mobile applications like augmented reality, healthcare sensing and analysis are not reaching their full potential since they are too compute intensive and too energy intensive. Applications like speech recognition [3]-[5], optical character recognition, language translators, image processors [6], [7], video processing [8], online games and wearable devices require powerful computational facilities. These applications require considerable energy consumption and good computing power which restricts the developers to implement them for mobile devices. When the computation intensive applications are tried on the handheld devices we must consider the factors like resource scarcity of mobile devices, large consumption of battery power and the network environment-unreliable, inconsistent and limited bandwidth network.

Among the various most promising biometric based approaches face recognition is the most popular one for recognizing people and is used widely in law enforcement and criminal investigations. Since human faces are meaningful, complex, multi-dimensional visual stimuli, it is very difficult to develop a computational model [9] for recognizing human face. Lot of research have been done in this area and as a result lot of solutions have been proposed for efficient face recognition [9]-[13]. Most of these solutions are applicable to desktop systems. When such an application has to be developed for a mobile device, we have to face many challenges because of the scarcity of resources of the mobile device. The battery life of a mobile device can be extended by selective offloading of resource-intensive execution. But the success depends on the reliability of end-to-end network [14]. At an event of natural calamity (earthquake, tsunami) or terrorist attack, the internet connectivity would have been destroyed. After few days or weeks, only limited connectivity may be re-established. But if we could use resource-intensive applications on mobile devices, that would be really helpful [14].

However, there are some major difficulties faced by mobile cloud computing. They include the WAN latency [15] and high energy consumption. In [16] Satyanarayan et al. proposed a solution known as cloudlet which is a new architectural element for cyber foraging that represents the middle tier of a 3-tier hierarchy: mobile device –cloudlet – cloud [17]. A cloudlet can support multiple VMs. This will help to reduce hardware dependencies on the cloudlet and between mobile device and cloudlet. The mobile device can carry as many overlays and would be able to execute on any cloudlet that has the same baseVM with which the application overlay was created. The setup time and administration time of the application can be considerably reduced by using the cloudlet with VM technology and it provides great flexibility by simplifying the deployment and configuration management.

This paper will provide some of the earlier works on face recognition, code offloading techniques and cloudlet provisioning mechanisms in Section 2. Section 3 will brief upon the VM Based Cloudlets and emphasize on the Dynamic VM Synthesis architecture. Section 4 will discuss two different approaches for implementing face recognition application using VM Based Cloudlet. Section 5 discusses the result. Finally, in Section 6 is the conclusion.

2. Related Works

Since our experiment is the integration of three different techniques-face recognition, code offloading and cloudlet provisioning, the related works are given in three different categories. The first category gives an idea about different face recognition techniques, the second category is about code offloading and the third category explains various cloudlet provisioning techniques.

2.1. Principal Component Analysis Based Face Recognition

Face recognition has a lot of practical applications like access control, design of human computer interface (HCI), automated crowd surveillance, criminal identification and so on. A large number of face recognition algorithms have been developed in last decades. These algorithms use different methods like PCA [7], [9], LDA [11], [18], ICA [19], [20], SVM [21], [22], and ANN [23], [24] for recognition.

The principal Component Analysis (PCA) [9] is one of the most popular method for face recognition which is also known as Eigen face method. This method finds a feature space with reduced dimension and then it is used for recognition. Even though it has complex calculations, it gives good results. Linear Discriminant Analysis (LDA) method proposed in [18] describes about feature selection. Most of the LDA based face recognition systems reduce dimensions using PCA and then LDA is used to maximize the discriminating power of feature selection for improving results. A combination of Gabor filter, PCA and LDA is used in [11]. Gabor filter is used to filter frontal face images and PCA is used for dimensionality reduction and then LDA is used for feature extraction. When different illumination conditions are considered PCA

gives good results than LDA.

Different methods for face detection and recognition are described in [10], [12], [13], [24], [25]. All these methods use PCA in one stage or another. Considering the advantages of PCA, we have used PCA Eigen face method for performing face recognition in our application.

2.2. Architecture for Code Offloading

Many architectures have been proposed in recent years for cloud offloading. Mobile Assistance Using Infrastructure (MAUI) [26] is one architecture for code offloading. It basically focusses on saving the mobile's energy consumption using remote execution. MObile Cloud Hybrid Architecture (MOCHA) [27] is another architecture for offloading the code. To overcome the latency problem with the cloud, this architecture uses special-purpose inexpensive compute-box with the capability of massively parallel processing and thus reducing the response time.

Cloudlets are a way to implement cyber foraging. There are many cloudlet architectures proposed. Here, we have studied VM Based Cloudlets [16]. These cloudlets are an effective way serving multiple users simultaneously in an isolated environment. Each user is allocated a separate VM specific to his application on the cloudlet. They follow the principle of "pre use customization and post use clean-up". VM Migration and Dynamic VM Synthesis are two major architectures for VM Based Cloudlets. VM Migration architecture pauses a running VM on mobile device, transfers it to the cloudlet and resumes it there. In other words, the memory image of the source server VM is transferred to the destination server without stopping execution [28]. An illusion of live and seamless migration is obtained by copying the memory pages of the VM priorly without affecting the operating system and other applications. Since the code is pre-copied, no code exchange is done during offloading. As far as the mobile device is concerned, VM Migration is heavy and it is time consuming.

Since the performance of dynamic VM Synthesis depends solely on local resources and WAN failure wouldn't affect synthesis, the authors propose to use dynamic VM synthesis. Dynamic VM Synthesis architecture [16], [29], [30] provisions a custom VM on the cloudlet dynamically and the mobile device discovers a cloudlet in its LAN network using any of the discovery protocols. Dynamic VM Synthesis architecture uses virtual machine technology to rapidly instantiate customized service software on a nearby cloudlet [16]. The mobile device acts as a thin client and can access this service over a Wireless LAN. When the mobile device establishes a TCP connection with the cloudlet and authenticates itself, then, it transfers a VM overlay to the cloudlet. The cloudlet applies this overlay to the base VM to generate the Launch VM running the backend server for the user application.

We have used a cloudlet for storing the huge database of images and the required resources. The face recognition application accesses the database of images and the application requires huge amount of processing. So the code and data are offloaded to the cloudlet from the mobile device. Then the cloudlet will do the required processing and it will give the results back to the mobile device. We will see VM based cloudlets in detail in the following section.

2.3. Cloudlet Provisioning Mechanisms

The client program is running on the mobile device and the ready to use server code will be on the Service VM. The process of configuring and deploying the service VM is known as VM Provisioning. Three different categories of cloudlet provisioning mechanisms are explained in [31]. They are run time, deployment time and on demand VM Provisioning.

2.3.1. Run time provisioning

Run time cloudlet provisioning is done from the mobile device [31]. VM Synthesis and Application virtualization are two different mechanisms for run time cloudlet provisioning. In VM Synthesis the mobile

device contain the application overlay as well as the client application and metadata. The cloudlet contains the exact base VM. When the mobile device sends the compressed overlay to the cloudlet server, the cloudlet server decompresses the overlay and it is applied to the baseVM to create the serviceVM. Then the serviceVM will be executed on the cloudlet server. The major advantage of VM synthesis is its flexibility. Any server code which can be installed on a base VM can be run on the cloudlet. But it requires the exact base VM to run the application.

In application virtualization [31] the server code is made as a self-contained application which can run independently. Here the mobile device contain the virtualized server code, the client application and the metadata. A matching VM is found on the cloudlet server based on the metadata send by the mobile device and then the virtualized application send by the mobile device will be deployed and started on a copy of the matching VM. VM synthesis is not required here, so the application ready time will be less compared to VM synthesis. Portability across OS is achieved due to virtualization but if any dependency is missed to capture, errors will occur.

2.3.2. Deployment time provisioning

When the cloudlets are preprovisioned based on mission needs, it comes under deployment time provisioning [31]. Cached VM and cloudlet push are two different mechanisms for deployment time provisioning. In cached VM the cloudlet server maintains a repository of VMs with various capabilities and each VM is considered as a service. The mobile device contains the client application and metadata. When the mobile device queries about a particular VM using the serviceVM ID and if that particular VM is found on the cloudlet server, then a copy of the service VM is started on the cloudlet server and the corresponding IP address and port number is send to the mobile client through which the client application can communicate with the server. It is possible to update the server code which is an advantage of this mechanism. The disadvantage is that the cloudlet server is provisioned by the service VMs required by the client applications.

In cloudlet push [31] the cloudlet server contains a repository of paired service VMs and client applications. When the mobile client selects a particular application from the repository, the cloudlet server will push the requested application that matches with the mobileOS to the mobile device. A copy of the service VM corresponding to the particular mobile client will be started on the cloudlet server and the IP address and port number of the server will be send to the mobile so that the client can communicate with the server. it supports most client mobile devices but the repository on the cloudlet server should contain client application that matches with the mobileOS.

2.3.3. On-demand provisioning

The capabilities for a cloudlet can be assembled at run time based on application demand and that kind of provisioning is known as on-demand VM provisioning [31]. In this approach the mobile device sends a provisioning script to the cloudlet server and then the serviceVM is assembled using a provisioning tool at run time according to the provisioning script. Then the server code will be started and the IP address and port number will be given to the client. Here the assembling of ServiceVM is done at run time but the cloudlet server should contain all required server code components.

3. Proposed Methods and Observations

Satyanarayan in [14] define cloudlet as a decentralized and self-managed resource rich computer with few users at a time. Many cloudlet architectures have been proposed by various researchers. We have studied VM Based Cloudlets [14] with Dynamic VM Synthesis [26]-[28] architecture and used it for hosting a novel face recognition technique. These cloudlets can be customized based on the application before their use and all the customizations are removed after their use.

A cloudlet in a network can be discovered using a discovery service by which the broadcasted IP address of the cloudlet is discovered using some discovery protocols like UPnP. Then the mobile can establish a connection with the cloudlet server using TCP or HTTP to transfer the overlay file to the cloudlet. After getting the overlay file it is applied on the base VM on the cloudlet and runs the server code for the particular application. VM overlay for the application has to be created in advance and stored on the device. It may be created on any machine with a base VM.

Our aim was to design a sample face recognition web application, implement it on cloudlet with Dynamic VM Synthesis architecture and measure the performance. For this, we used Elijah Provisioning, a GitHub project [32] which helped us to provision the custom VM and to do Dynamic VM Synthesis [30]. We conducted two different experiments on a cloudlet with Ubuntu 14.04 as host and Ubuntu 12.04 as base VM. The first approach is a web application for face recognition on the cloudlet using apache tomcat server on the Virtual machine and the second approach uses TCP sockets for communication between mobile client and cloudlet server.

3.1. Approach 1: Face Recognition as a Web Application Using Cloudlet

In this approach [33] the mobile device acts as a thin client which will transfer the captured image to the cloudlet server. All other processing required for face recognition is done on the cloudlet server and finally the mobile device gets the result from the cloudlet server.

Our face recognition application follows skin color based face extraction and then the method of Principal Component Analysis [9] for face recognition and is deployed on Apache Tomcat Server. The database used in our experiment is the FERET color database which contain frontal face images. The exact face region is found out by skin color detection using YCbCr (Y is the luma component and Cb and Cr are the blue-difference and red-difference chroma components) color space from both the training set images and test image. The YCbCr color space is applied to the training and test images for getting the skin images and then suitable threshold is applied to get binary images. After that morphological operations like opening and closing are performed for removing noise and holes in the skin image. Morphological opening is erosion followed by dilation and morphological closing is dilation followed by erosion. For extracting the face region from each image initially we are finding the contours in the skin image. After that the blob corresponding to the face region in the skin image is found out by iterating over the contours to get the exact face region. Then the face recognition is done by using PCA Eigen face method.

Fig. 1. Training set.

Fig. 2. Skin region from training set.

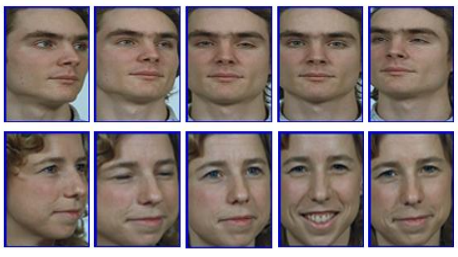

Fig. 3. Extracted Face Region using skin color information.

The main idea of Principal Component Analysis (PCA) is to find the vectors that best account for the distribution of face images within the entire image space. A face image $I(x, y)$ can be considered as a two dimensional $N \times N$ array of intensity values. When we convert this image to a vector of dimension N^2, it becomes a point in the N^2 dimensional space. For example, a 256×256 image becomes a vector of dimension 65,536 or a point in 65,536-dimensional space. The PCA method [9] can be explained step by step as below. Consider a training set consisting of total M images.

$$S = \{I_1, I_2, I_3, \ldots\ldots\ldots\ldots, I_M\}$$

1. Convert face images in the training set to face vectors. Now

$$S = \{\Gamma_1, \Gamma_2, \Gamma_3, \ldots\ldots\ldots, \Gamma_M\}$$

2. The next step is normalization of face vectors. For that
   i. Find the average face vector ψ

$$\Psi = \frac{1}{M}\sum_{n=1}^{M} \Gamma_n \quad (1)$$

   ii. Subtract the average face vector from each face vector

$$\Phi_i = \Gamma_i - \psi \quad (2)$$

3. Reduce the dimensionality of the training set by calculating the covariance matrix as

$$C = A^T A \quad \text{where } A = [\Phi_1\, \Phi_2\, \Phi_3 \ldots\ldots\ldots \Phi_M] \text{ of dimension } N^2 \times M \quad (3)$$

So $C = M \times N^2 \,.\, N^2 \times M = M \times M$ where $M << N^2$

4. Find the Eigen vectors from the new covariance matrix

If v_i are the eigenvectors of $A^T A$ such that

$$A^T A \; v_i = \mu_i v_i$$

Premultiplying both sides by A, we get $AA^T A \; v_i = \mu_i A v_i$

From which we see that Av_i are the eigenvectors of $C = A^T A$

$$u_i = A v_i = \sum_{k=1}^{M} v_{ik} \Gamma_k \quad (4)$$

where μ_i are the eigenvalues and Av_i are eigenvectors of AA^T

5. Select K eigenvectors such that $K<M$ and can represent the whole training set. The selected K eigenvectors must be in the original dimensionality of the face vector space.
6. Convert the lower dimensional K eigenvectors to original face dimensionality.
7. Represent each image as a linear combination of all K eigenvectors. Each face from training set can be represented as the weighted sum of K eigenfaces + mean face. The weight vector $\Omega^T = [w_1\, w_2\, w_3 \ldots\ldots w_K]$ describes the contribution of each eigenface in representing an image.
8. A new face image (Γ) is transformed into its eigenface components as

$$w_k = u_k^T (\Gamma - \psi) \qquad \text{where } k = 1..K. \quad (5)$$

9. Calculate the Euclidian distance between the input weight vector and all the weight vectors of training set as

$$\epsilon_k^2 = \| \Omega - \Omega_k{}^2 \| \quad (6)$$

A face is classified to class k if the minimum ϵ_k is below some threshold. Otherwise it is classified as unknown.

During the face recognition phase, the extracted face images which is the training set is converted into face vectors. Then the face vectors are normalized by finding the average face and by subtracting the average face from each training image. A new covariance matrix is found out to reduce the dimensionality and then Eigen vectors are calculated from this covariance matrix. Only K (K < number of images in the

training set) eigenvectors are selected such that they can represent the whole training set. These K Eigen vectors are then converted to original dimension and all the training set images are represented as a linear combination of the K eigenvectors with a weight vector for each image.

When an unknown face comes for recognition, it is converted into a face vector. After normalizing the face vector, the normalized face vector is projected on to the face space and a weight vector is found out. Based on the Euclidean distance between the weight vectors of input image and training images, a confidence level is found out and the image is recognized based on this confidence level.

Fig. 4. Average face image.

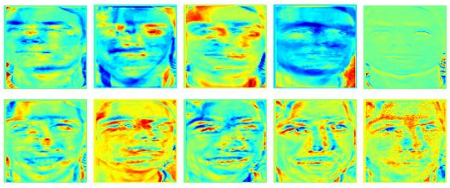

Fig. 5. Eigen faces.

3.1.1. How the mobile device finds a cloudlet?

The UpnP architecture allows device-to-device networking of consumer electronics, personal computers, mobile devices and networked home appliances. The UpnP architecture supports without configuration of networking. A UpnP compatible device from any vendor can dynamically connect to a network, obtain an IP address, announce its name, advertise or convey its capabilities upon request, and it also allows to learn about the presence and capabilities of other devices in the vicinity. Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) servers are optional and are only used if they are available on the network. Devices can disconnect from the network automatically without leaving state information. The foundation for UpnP networking is IP addressing. A DHCP client and search for a DHCP server must be implemented in each device, when the device is first connected to the network. If no DHCP server is available, the device must assign itself an address. The process of assigning an IP address for itself is called within the UpnP Device Architecture as Auto IP. Once a device has acquired an IP address, the following step in UpnP networking is discovery. The UpnP discovery protocol is known as the Simple Service Discovery Protocol (SSDP). When a device is added to the network, SSDP allows that device to publicize its services to control points on the network. This is obtained by sending SSDP alive messages. When a control point is added to the network, SSDP allows that control point to actively search for instruments of interest on the network or listen passively to the SSDP alive messages of device.

3.1.2. VM synthesis

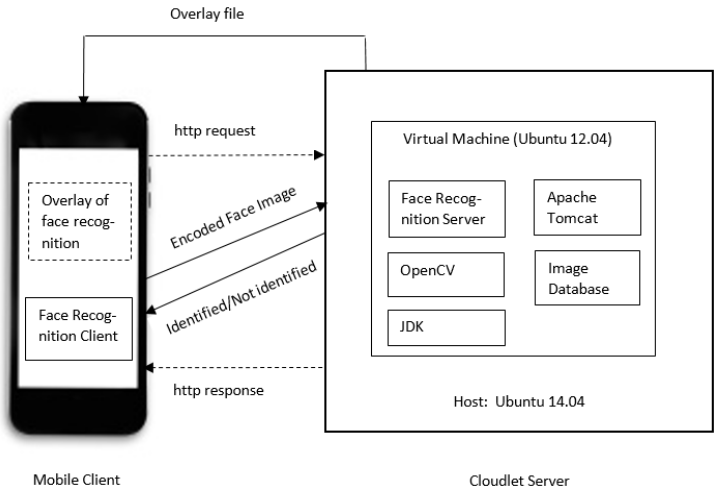

Fig. 6. Block diagram of the face recognition web application using cloudlet.

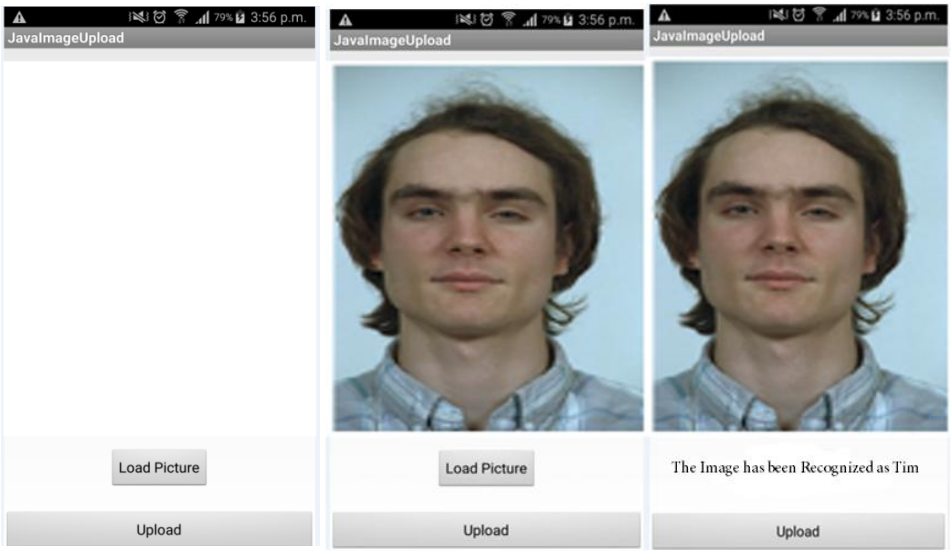

Fig. 7. Screenshots of mobile user interface for sending the image and receiving the output.

As shown in Fig. 6, in the cloudlet server the virtual machine is customized with all the necessary resources for the face recognition application. After all the customization, the overlay file is created and stored on the cloudlet server. This overlay file can be copied to the mobile device. Now when the mobile device comes in the vicinity of the cloudlet under the same network, the mobile device will get the IP address of the cloudlet server which is broadcasted the cloudlet server. Then the mobile device establishes connection with the cloudlet server and the overlay file along with metadata is send to the cloudlet server. The cloudlet server will synthesizes the VM so that the face recognition runs in the virtual machine. For this application the mobile device acts as a thin client which captures a face image and send to the cloudlet for recognition.

When the face recognition server gets the input test image, it starts the process of recognition and the result is given back to the mobile device.

In short, following is the step by step procedure we followed for our experiment.

1. The VM in the Cloudlet server is made ready with JDK, OpenCV, Apache Tomcat, Image Database and Java Face Recognition Server
2. An overlay file is created with the changes made in the VM on the Cloudlet Server
3. The overlay file is copied to the mobile client
4. The mobile client stores the overlay file. This overlay file can also be used for further use.
5. The face image is encoded to string
6. HTTP call is made to upload the encoded image on the Cloudlet Server
7. The overlay file along with metadata is given to the Cloudlet Server
8. On the Cloudlet Server the image is decoded and stored
9. The overlay synthesis is done on the Cloudlet Server and the Face Recognition Server code is executed on the VM
10. The result is given back to the mobile client.

3.1.3. Observations

The compressed overlay file created for the face recognition application is of 600 MB, VM synthesis is getting delayed which is not desirable for applications that require real time response. If we use a cloudlet with more computing speed, we can reduce the delay in response time. And also, our application used Apache web server and HTTP connection for accomplishing the tasks. We can eliminate the use of web server if we go with socket communication between mobile device and application server and thus the size

of overlay file can be reduced.

In our experiment the face recognition application using PCA gave an accuracy of 92% which is improved to 95% when the face extraction algorithm using skin color is added with this.

3.2. Approach 2: Real Time Face Detection and Recognition Using Cloudlet

In this approach the mobile device is not a thin client. It captures the image and some preprocessing is done in the mobile device itself. The mobile device will detect the faces in the image and then crops the face. Now the cropped face will be send to the cloudlet server. The recognition process will be done on the server and the result will be given back to the mobile device.

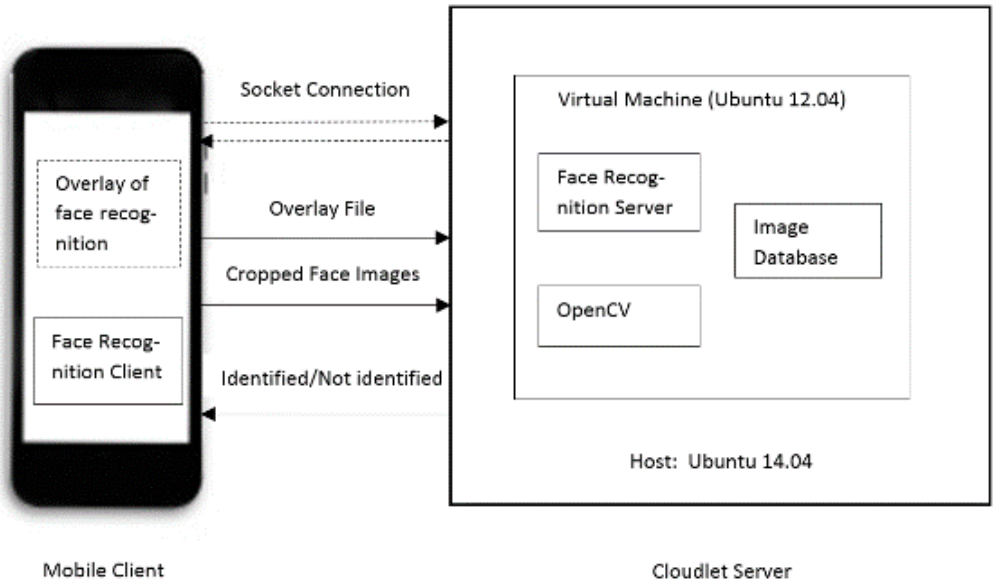

Fig. 8. Block diagram of real time face recognition using cloudlet.

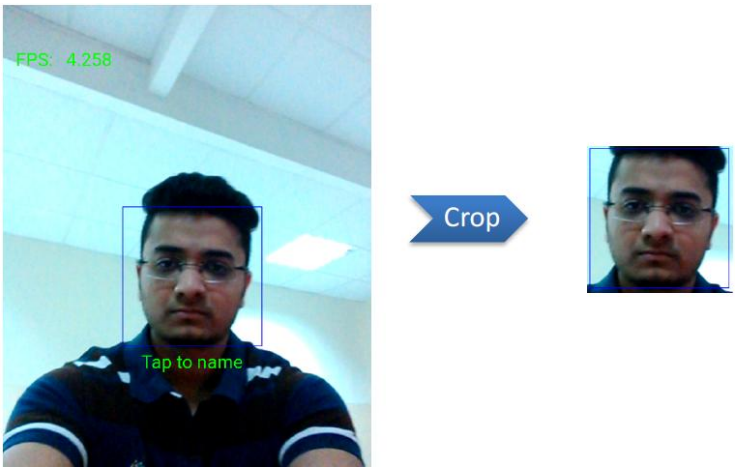

Fig. 9. Face Detection and cropping on mobile device.

In this experiment we have established connection of the mobile device to the cloudlet using Socket connection. For this purpose, a file transfer code was written in Java and executed on the computer. The server code opens a server socket and listens to incoming connection from the client socket. When the mobile client and cloudlet server are successfully connected, the cloudlet server receives the incoming file from the client application. The client application installed on a mobile device in our case is a simple Android application which connects the mobile device with the cloudlet server whose IP address has already been discovered. Then the overlay file can be transferred to the cloudlet using this socket

connection. Once the overlay file has been transferred to the cloudlet, the cloudlet node synthesizes the overlay file and loads it onto the Base VM to resume the VM in its suspended state. Then the face recognition server will be executed in the virtual machine. The mobile client then sends the cropped face image to the face recognition server. In this approach also we have used PCA based face recognition. The server recognizes the face and the result is given back to the mobile device.

Fig. 10. Screenshot showing IP address of cloudlet on mobile device. Using the "connect" button the mobile device and cloudlet server will be connected and the overlay file will be transferred to the cloudlet server.

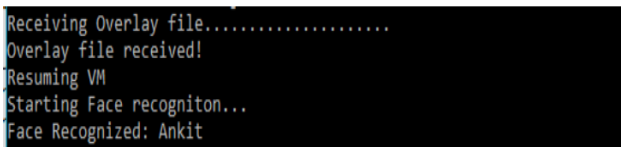

Fig. 11. Face Recognition result on the cloudlet.

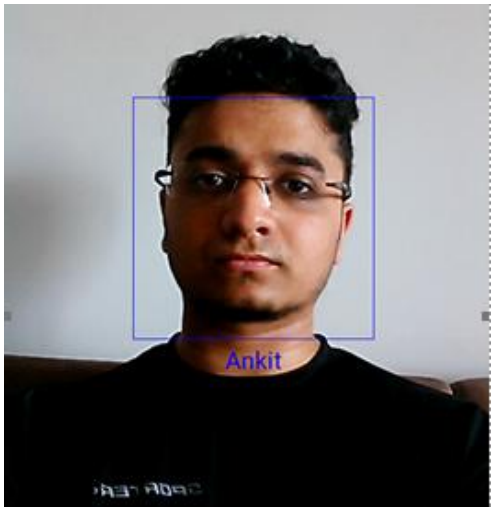

Fig. 12. Face Recognition result on the mobile.

In short, following is the step by step procedure we followed for our experiment.

1. The cloudlet server is made ready with OpenCV, Image Database and C++ face recognition server

2. An overlay file is created with the changes made in the BaseVM
3. The compressed overlay file is copied to the mobile client
4. The mobile client stores the overlay file. This overlay file can be used whenever needed.
5. When the client want to recognize a particular person, take a photo using mobile.
6. The face recognition client on the mobile device will detect and crop the face from the image.
7. The cloudlet client on the mobile device will find the nearby cloudlet server.
8. The mobile device and the cloudlet server will be connected using socket connection
9. The overlay file and the cropped face image will be send to the cloudlet server.
10. The overlay synthesis is done on the Cloudlet Server and the Face Recognition Server code is executed on the VM
11. The result is given back to the mobile client.

4. Results

We have shown the two different approaches experimented for face recognition on cloudlet. The face recognition application has been implemented on VM Based Cloudlets [16] with Dynamic VM Synthesis [28]-[30] architecture. The configuration of mobile client and cloudlet server we used for experiment are shown below.

Table 1. Mobile – Cloudlet Configuration

	Mobile client	Cloudlet
	Samsung Galaxy S5	Desktop/ Laptop
CPU	Quad-core 2.5 GHz Krait 400	Intel Core-i3/i5/i7
GPU	Adreno 330	
Internal Memory	32 GB 2 GB RAM	4GB/8GB/ 16GB
WiFi	802.11 a/b/g/n/ac	802.11 ac

For Approach 1 we used three different cloudlet severs with different configuration due to the bigger size of the overlay file. First we tried the application on a cloudlet with 4GB RAM. Since the overlay size was 664MB, the overlay transmission time as well as the overlay synthesis time was found to be very large. It took nearly 4 hours for overlay synthesis. Then we tried with a cloudlet server with 8GB RAM and in that case overlay synthesis took nearly 2 hours. Finally we tried it with a cloudlet server having 16GB RAM and the overlay synthesis took only 15.49 seconds. So we found that we have to reduce the overlay size for the application. If the overlay size is bigger we need a cloudlet server with high computing power. Also the energy consumption of the mobile device for the application depends on the size of the overlay file. The battery consumption of the mobile device will be high if the size of the overlay file is high.

Table 2. Storage Requirement

	Base VM Size (MB)	Overlay Size (MB)	Zip Overhead (bytes)
Approach 1	730.6	664.7	251
Approach 2	730.6	36.3	251

Table 3. Time Requirement in Seconds

	Overlay Upload	Overlay Decompression	VM Synthesis
Approach 1	135.34	34.62	15.49
Approach 2	79.26	15.78	5.24

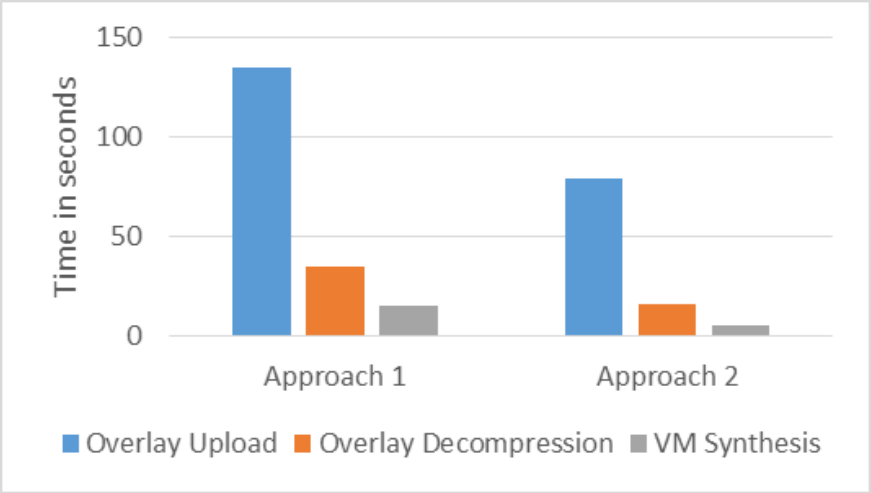

Fig. 13. Time taken by each approach.

5. Conclusion

Cloudlet is a new architectural element for cyber foraging that represents the middle tier of a 3-tier hierarchy: mobile device –cloudlet – cloud. A cloudlet can support multiple VMs. This will help to reduce hardware dependencies on the cloudlet and between mobile device and cloudlet. The mobile device can carry as many overlays and would be able to execute on any cloudlet that has the same baseVM with which the application overlay was created. The setup time and administration time of the application can be considerably reduced by using the cloudlet with VM technology and it provides great flexibility by simplifying the deployment and configuration management. We have explained two different approaches for face recognition on mobile devices using cloudlet. Cloudlet based biometric applications for mobile devices will be useful at the time of natural calamities for identifying people where end-to-end network connectivity is a problem. It can also be used for law enforcement or for criminal investigations at places like railway station, airports etc. as it gives real time response.

Acknowledgment

The authors wish to acknowledge and appreciate the GitHub Project, Elijah Cloudlet by Carnegie Mellon University as it played a key role in our studies.

References

[1] Satyanarayanan, M., Chen, Z., Ha, K., & Hu, W. (2014). Cloudlets: At the leading edge of mobile-cloud convergence. *Proceedings of 6th International Conference on Mobile Computing, Applications and Services (MobiCASE)*, Austin, TX.

[2] Weiser, M. (1999). The computer for the 21st century. *Newsletter-ACM SIGMOBILE Mobile Computing and Communications Review*, *3(3)*, 3-11.

[3] Flinn, J., Park, S,. & Satyanarayanan, M. (2002). Balancing performance, energy, and quality in pervasive computing. *Proceedings of 22nd IEEE International Conference on Distributed Computing Systems*.

[4] Su, Y.-Y., & Flinn, J. (2005). Slingshot: Deploying stateful services in wireless hotspots. *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, Seattle, WA, ACM.

[5] Balan, R., K., Gergle, D., Satyanarayanan, M., & Herbsleb, J. (2007). Simplifying cyber foraging for mobile devices. *Proceedings of the 5th International Conference on Mobile Systems, Applications and Services*.

[6] Kristense, M. D., & Bouvin, N. O. (2008). Developing cyber foraging applications for portable devices. *Proceedings of 7th IEEE Conference on Polymers and Adhesives in Microelectronics and Photonics and 2nd*

IEEE International Interdisciplinary Conference on Portable Information Devices, PORTABLE-POLYTRONIC, Garmish –Partenkirchen.

[7] Porras, J., Riva, O., & Kristensen, M. D. (2009). Dynamic resource management and cyber foraging. *Middleware for Network Eccentric and Mobile Applications, Springer Berlin Heidelberg*, 349-368.

[8] Chun, B.-G., & Maniatis, P. (2009). Augmented smartphone applications through clone cloud execution. *Proceedings of the 12th conference on Hot Topics in Operating Systems, HotOS'09*.

[9] Turk, & Pentland, A. (1991). Eigenfaces for recognition. *ACM Journal of Cognitive Neuroscience*, *3(1)*, 71-86.

[10] Luo, L., Swamy, M. S., & Plotkin. E. I. (2003). A modified PCA algorithm for face recognition. *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*.

[11] Liu, C., & Wechsler, H. (2002). Gabor feature based Classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing*, *11(4)*, 467-476.

[12] Han, Y-H., & Kwak, K. C. (2010). Face recognition and representation by tensor-based MPCA approach. *Proceedings of the 3rd International Conference on Machine Vision*.

[13] Jadhav, I., S., Gaikwad, V., T., & Patil, G., U. (2011). Human identification using face and voice recognition. *International Journal of Computer Science and Information Technologies*, *2(3),* 1248-1252.

[14] Ha, K., Lewis, G., Simanta, S., & Satyanarayanan, M. (2011). *Cloud Offload in Hostile Environments.* Carnegie Mellon University, Pittsburgh.

[15] Huang, J., Xu, Q., Tiwana, B., Mao, Z., M., Zhang, M., & Bahl, P. (2010). Anatomizing application performance differences on smartphones. *Proceedings of the 8th international conference on Mobile systems, applications, MobiSys '10*, San Francisco, CA.

[16] Satyanarayanan, M., Bahl, P., Caceres, R. & Davies, N. (2009). The case for VM-based cloudlets in mobile computing. *Pervasive Computing* , *8(4)*, 14-23.

[17] Ha, K., Pillai, P., Richter, W., Abe, Y., & Satyanarayanan, M. (2013). Just-in-time provisioning for cyber foraging. *Proceeding of the 11th annual international conference on Mobile Systems, Applications, and Services-MobiSys '13*, Taipei, Taiwan.

[18] Swets, D. L., & Weng, J. (2002). Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *18(8)*, 831-836

[19] Lei, J., Lu, C., & Pan, Z. (2011). Enhancement of Components in ICA for face recognition. *Proceedings of the 9th International Conference on Software Engineering Research, Management and Applications (SERA),* Baltimore, MD.

[20] Sejnowski, T. J., *et al*. (2002). Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, *13(6)*, 1450–1464.

[21] Guo, G., Li, S. Z. & Chan, K. L. (2000). Face recognition by support vector machines. *Proceedings of Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble.

[22] Phillips, P. J. (1999). *Support Vector Machines Applied to Face Recognition*, MIT Press.

[23] Omar, N. N. S. A., & Khalid, M. (2007). Face recognition system using artificial neural networks approach. *Proceedings of the International Conference on Signal Processing, Communications and Networking, ICSCN '07*, Chennai.

[24] Siddiky, F. A., Alam, M. S., Ahsan, T., & Rahim, M. S. (2007). An efficient approach to rotation invariant face detection using PCA, generalized regression neural network and Mahalanobis distance by reducing search space. *Proceedings of the 10th International Conference on Computer and Information Technology,* Dhaka.

[25] Mittal, N., & Walia, E. (2008). Face recognition using improved fast PCA algorithm. *Congress on Image and Signal Processing*, Sanya, Hainan.

[26] Cho, D., Wolman, A., Saroiu, S., Chandra, R., & Bahl, P. (2010). MAUI: Making smartphones last longer with code offload. *Proceedings of the 8th International ACM Conference on Mobile Systems, Applications, and Services*.

[27] Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., & Heinzelman, W. (2012). Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. *Proceedings of IEEE Symposium on Computers and Communications (ISCC).*

[28] Teka, F. A. (2014). *Seamless Live Virtual Machine Migration for Cloudlet Users with Multipath TCP*. Carleton University, Ottawa, Ontario.

[29] Tactical Cloudlets: Moving Cloud Computing to the Edge. From https://www.sei.cmu.edu/mobilecomputing/research/tactical-cloudlets/

[30] Wolbach, A., Harkes, J., Chellappa, S., & Satyanarayanan, M. (2008). Transient customization of mobile computing infrastructure. *Proceedings of the First Workshop on Virtualization in Mobile Computing*.

[31] Echeverria, S., Root, J., Bradshaw, B., & Lewis, G. (2014). On-demand VM provisioning for cloudlet-based cyber-foraging in resource-constrained environments. *Proceedings of the 6th International Conference on Mobile Computing, Applications and Services.*

[32] GITHUB. From https://github.com/cmusatyalab/e14lijah-cloudlet

[33] Praseetha, V. M., & Vadivel, S. (2016). Face extraction using skin color and PCA face recognition in a mobile cloudlet environment. *Proceedings of the 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 41-45, Oxford, UK.

Praseetha V. M. received her M.Tech in computer and information science from Cochin University of Science and Technology, Kerala, India. She is a Ph.D. student affiliated with the Department of Computer Science BITS Pilani, Dubai Campus. Her main research interests are mobile cloud computing and distributed computing.

Ankit Bansal was born in 1993 and got the bachelor degree in the Birla Institute of Technology and Science (BITS), Pilani in Dubai, UAE. His major is in computer science and engineering. His research interests include software systems, cloud computing and artificial intelligence. He is pursuing master's in computer science in University of Michigan in Ann Arbor, USA.

S. Vadivel is a professor at the Department of Computer Science, BITS Pilani, Dubai Campus. He completed his Ph. D at IIT Madras. His research areas cover a wide range of advanced topics including Cloud computing, Web services and service oriented architecture, data mining, mobile computing, and software engineering applied to Service oriented architecture.