Chapter 3

Architecture Based Methodology to Extend the Energy of Mobile Devices

3.1 INTRODUCTION

Currently, in Cloud Computing (Fortis et al., 2015) mobile subscribers are supplanting their general seniority cell telephone to the recently skilled advanced mobile phones. At one time, the general mobile devices were utilized just to make calls or send messages. They could not be utilized in application development. Accordingly, the constant backing could not be attained. Now, the development innovation of the web and multimedia has concocted very competent devices with enough preparation ability, an effective battery and other such gimmicks in one small device.

As mobile devices are made to handle mixtures of programming uses, they are discharged with less supervision efforts or connection by the administration suppliers. Mobile devices are made for using cloud computing services, and their focal points with the joining of the mobile environment with cloud computing will structure Mobile Cloud Computing (MCC). With this reconciliation, today’s Mobile Devices feature the most recent type of cloud services and the cloud capacities for mobile device supporters.

 Appearing differently in relation to the prior innovation of mobile computing, the MCC is situated as an assortment of appropriated servers and workstations, for which resources are totally virtualized and can be utilized at whatever point required as a part of any sort of structure. There are numerous applications focused around MCC; for example, Gmail, Google Maps, Apple’s MobileMe, Microsoft Live Mesh, and Motorola’s MOTOBLUR are extremely well-known applications and have been generally utilized by their endorsers (Schuring, 2011).

Numerous applications are too excessively computationally intensive to do on a mobile device. In the event that a versatile client needs to utilize such applications, the execution should be performed in the cloud. Different applications (Tawalbeh et al., 2015), such as picture recovery, voice distinction, gaming, and route determination could run on a mobile device. Nevertheless, they devour huge amounts of vitality, low-power planning has been a dynamic examination subject for a long time (Xiao and Han, 2014). There had been several research outcomes in the areas of energy preser-
vation of electronic devices (Gadallah and Kunz, 2006; Ghada et al., 2010; Hsu et al., 2011; Benkhelifa et al., 2015). In IEE Explore, searching for “low” and “power” in the archive title delivers more than 5000 outcomes. There are four essential methodologies to sparing vitality and expanding the battery life of mobile devices by evaluating them (Kumar and Lu, 2010; Hu et al., 2007).

**Adopt a new generation of semiconductor technology:** As transistors become tinier, every transistor devours less power. Tragically, as transistors become slighter, more transistors are expected to be utilized, providing further functionalities and better execution; therefore, energy consumption really increases.

**Abstain from wasting energy:** Entire frameworks or separate segments might enter stand-in or slumber modes to spare energy.

**Execute programs slowly:** When a system processor’s clock velocity twice, the power utilization almost octuples. On the off chance that the system clock speed is halved, the computation time twice; however, only a one fourth of the vitality is expended.

**Eliminate computation altogether:** The mobile device does not do the computation; rather, the computation is performed someplace else, consequently developing the battery lifetime of the mobile system.

### 3.1.1 OFFLOADING COMPUTATION TO SAVE ENERGY

Sending execution to an additional system is not a unique thought. Right now, the famous client-server method empowers mobile clients to dispatch browsers, search the web, and do shopping on the web. What discriminates cloud computing from the current method is the concept of virtualization. Rather than service providers control programs executing on remove servers, virtualization allows cloud suppliers to execute self-assertive applications from diverse users on virtual machines.

Computation offloading is a system that moves resource-intensive calculations from a local mobile system to the asset rich cloud (called the close-by framework). Cloud-based computation offloading upgrades applications’ execution (Jaradat et al., 2015), decreases battery power utilization, and executes applications that were not previously able to execute because of the lack of Mobile devices resources. In addition, the cloud offers storage services (Amazon, 2015).

Since offloading moves execution to a additional assetful system, it includes making a choice with respect to what and whether computation to move. An incomprehensible group of research exists on offloading choices for saving energy.

### 3.1.2 COMPUTATION OFFLOADING DECISION CRITERIA

Computation Offloading has two primary classes of choice: 1) what computation has to be offloaded and 2) where to offload. The computation to offload choice is accom-
plished by utilizing the partitioning approach, where the application that requires over-whelming computation is separated in two sections: 1) the one that might be processed locally and 2) the one that is possible for offloading. The where computation to offload choice decides the location in which the computation will occur. The vast majority of the systems and research studies on the spotlight of the binary choice issue, where they simply consider two alternatives: 1) to offload and 2) not to offload; in this methodology, a client views the cloud as a solitary element. Then again, there is the M-ary choice issue, where the client considers offloading and perceives more than one cloud element (e.g. a few cloud assets). This methodology turns out to be more intricate, since the client has more estimation to perform with a specific end goal to settle on a decent choice; a client needs to assess all and each conceivable accessible offloading location. A computation can occur on various sorts of substances (e.g. mobile device, cloudlet, cloud) as said in the past area. To make a computation offloading choice, it is critical to assess the trade-offs between communication and computation. The client needs to estimate the time for local computation and the time for remote computation that incorporates an ideal opportunity to trade the input and output information between the mobile device and the surrogate (i.e. cloud asset). Computation offloading gets to be beneficial at whatever point a program requires substantial computation and the computing resources are essentially quicker than the local computing assets. Likewise, when there is high bandwidth accessible and/or just a slight amount of information should be exchanged between the client and the server. The goals of computation offloading include: 1) to minimize the computation finish time 2) to shift the energy consumption to a more resource-rich facility and 3) to optimize the cloud computing asset allotment.

Computation Offloading is intricate since a few components should be considered (Sharifi et al., 2012; Sookhak et al., 2014). Whether the offloading is static or dynamic, choosing where to offload (e.g. cloud, cloudlet, mobile device), picking a solitary surrogate or numerous surrogates, distinguishing the area of the work and the input information to be handled (i.e. as of now in the cloud or gave by the client). Likewise, it is imperative to consider the surrogate attributes (e.g. memory limit, CPU speed, storage limit, responsiveness, and so on.), and if the mobile device will move the entire application or just some portion of it. A Computation Offloading Taxonomy is appeared in Figure 3.1.

There are a few criteria that impact the computation offloading choice and are sorted out in a few sets (Sharifi et al., 2012; Gani et al., 2014). The first set of criteria are related to the Cloud Service Provider (CSP) contemplations that assess the operation costs, the accessible resources, the income, and the location. The second arrangement of criteria are identified with User Preferences, where the user considers the expense of utilizing a remote resources, the security of the cloud, the reliabilty of the servers, and the responsiveness of the service. The third arrangement of criteria are identified with
communication network conditions such as available bandwidth, link quality, congestion, and number of hops to reach a cloud resource. The fourth arrangement of criteria are identified with the Surrogate Characteristics such as CPU speed, memory limit, storage limit, current system load, and accessible resources on the servers. At long last, the fifth arrangement of criteria are identified with the Computation Job Characteristics such as the size of the computation, the measure of the input data, and the span of the output information.

3.2 SAVE ENERGY

Smartphones are no more used just for voice correspondence; rather, they are used for gaming, watching videos, web surfing, and numerous different purposes. Thus, these frameworks would probably expend more energy and shorten the battery lifetime. Despite the fact that battery innovation has been consistently enhancing, yet it has not possessed the capacity to keep up with the fast development of energy utilization of these mobile devices. Offloading may save battery lifetime by moving the energy-intensive fragments of the calculation to remote servers (Kumar and Lu, 2010).

3.3 RELATED WORK

Energy utilization has been and will be one of the greatest worries in mobile/ wireless networks. Keeping in mind the ultimate goal to have the mobility, mobile devices depend on battery power, and moreover are constrained by the battery limit regardless of the dramatic change in functionality and hardware infrastructure. Understanding the energy consumption of mobile systems is vital for further diminishing it. To uncover the attributes of energy cost in mobile systems, researchers have performed estimation
studies that give huge experiences from various aspects of mobile computing.

(Zhang et. al, 2016) researched about performance and energy efficiency of migrating video applications to the cloud under dynamic remote network channels using state-of-the-art mobile platforms. Based on the distinguished difficulties, cloud opens doors for offloading realtime video applications, they planned a common energy-efficient offloading scheduling issue and proposed a adaptive scheduling algorithm that makes fine-grained offloading choices as per the dynamic wireless network constraints. They further assessed the adequacy of our solution through trace-driven simulations also, extensive experiments. At last, they exhibited two case studies on video cloud gaming and mobile remote desktop access to assess the performance of their solution in realworld video applications.

Several works exists for offloading applications to the cloud for execution and get the result to the mobile system. Authors(Kosta et al., 2012; Cuervo et al., 2010) have come out with offloading techniques that consider either of the factors like resource utilisation history or future consumption demand, leading the offloading system to several limitations and expensive computational cost. The novelty of the proposed system lies in the fact that it considers not only resource utilisation history and future consumption demand, but takes offloading decision based on collective consideration of factors like delay tolerance, favourable network bandwidth, cloud capacity and energy consumption. This makes the proposed framework effective when compared to the traditional offloading framework.

3.4 OBJECTIVE

The objective of this chapter is to provide an architecture to extend the energy of mobile devices using Amazon Web Services.

3.5 THE PROPOSED METHODOLOGY

In this section we discuss about computation offloading decision for saving energy, proposed architecture for offloading, algorithm and experimental setup

3.6 ANALYSIS OF ENERGY FOR OFFLOADING

Computation offloading has two primary objectives: 1) diminish the energy consumption on mobile devices, and 2) diminish execution time of applications (Kumar et al., 2013). The center of this exposition is to extend the vitality of mobile system. At whatever point mobile device clients encounter a circumstance in which they can’t execute an application inside certain time imperatives, they should seriously mull over computation offloading. In any such case, a choice examination must be performed before
making any move. A trade-off between the communication and computation delays needs to be considered, as represented in Figure 3.2. In common, computation offloading is advantageous whenever: 1) a job contains a lot of computation 2) just a little measure of data needs to be traded between the mobile device and the cloud resource and 3) when high network bandwidth is accessible (Kumar et al., 2013; Kumar and Lu, 2010). At some point there is a lot of information to transmit and the computation is little, computation offloading turns out to be less alluring and significantly more costly than utilizing local assets. In different cases, the computation offloading choice relies on upon the accessible bandwidth to trade information between substances; with high bandwidth, computation offloading might be a good choice. Consequently, it is critical to consider the attributes of: 1) the job 2) the available resources and 3) the network. These elements impact the proportion between computation time and communication time and finally figure out whether computation offloading is helpful.

Figure 3.2 Trade off between computation and communication

For the offloading services, the vitality usage for offloading, ought not be more prominent than that of local execution (Jaradat et al., 2015). The following examination clarifies the conditions when offloading spares energy.

Give $S_m$ a chance to be the speed of the mobile device. Assume $P_c$ is the sum of computation for the offloaded segment and $P_m$ is the energy on the mobile system. The energy (Kumar and Lu, 2010) to play out the assignment can be gotten

$$P_m \times \frac{w}{S_m}$$  \hspace{1cm} (3.1)

Give $P_s$ a chance to be the energy needed to send information from the mobile device through the network. In the wake of sending the information, the framework
requires to survey the device network interface while waiting for the consequence of
the offloaded computation. Amid this time, the energy utilization is $P_i$. The energy to
offload and execute the offloaded part is

$$P_c \times \frac{B}{BW} + P_i \times \frac{P_c}{S_c}$$  \hspace{1cm} (3.2)

- large $P_c$: the program needs substantial computation.
- large $S_c$: the server is fast.
- small $B$: a little amount of data is traded.
- large $BW$: the bandwidth is high.

Offloading saves energy when

$$P_m \times \frac{P_c}{S_m} > P_c \times \frac{B}{BW} + P_i \times \frac{P_c}{S_c}$$  \hspace{1cm} (3.3)

$$P_m \times \frac{P_c}{S_m} - P_i \times \frac{P_c}{S_c} > P_c \times \frac{B}{BW}$$  \hspace{1cm} (3.4)

$$P_c \times \left(\frac{P_c}{S_m} - \frac{P_c}{S_c}\right) > P_c \times \frac{B}{BW}$$  \hspace{1cm} (3.5)

From the analysis the amount of energy saving is

$$P_c \times \frac{E}{S_m} - P_i \times \frac{E}{S_c} - P_{sr} \times \frac{B}{BW}$$  \hspace{1cm} (3.6)

- Suppose the computation requires $E$ instructions
- $P_c$ - The mobile system consumes for computing.
- $P_i$ - The mobile system consumes while being idle.
- $P_{sr}$ - The mobile system consumes for sending and receiving data.
- $S_m$ - The speeds in instructions per second of the mobile system.
- $S_c$ - The speeds in instructions per second of the cloud server
- $B$ - The bytes which the server and mobile system exchanged
- $BW$ - The network Bandwidth

If the server is $F$ times faster ($S_c = F \times S_m$) then 3.6 becomes

$$\frac{E}{S_m} \times (P_c - \frac{P_i}{F}) - P_{sr} \times \frac{B}{BW}$$  \hspace{1cm} (3.7)
3.6.1 CPU WORKLOAD PREDICTION

(Park et. al., 2014) proposed an innocent history-based strategy to calculate the average execution time of an application on smartphone. It influences typical CPU workload that it got from the asset monitor and input data size of the application to anticipate processing time utilizing the history log L.

Based on the CPU workload forecast, the asset monitor utilizes the fundamental Exponential Moving Average algorithm (EMA for short) (Urgstahler and Neubauer, 2002) to foresee the average CPU load of smartphone. It stores the CPU load $cw_t$ in database as (Timestamp, CPU workload) periodically. Given current time period $t$, the EMA esteem for CPU workload is ascertained recursively by Equations 3.8, 3.9 and 3.10, where $CW_t$ is the estimation of the EMA at any time period $t$, coefficient $\delta$ represents the level of weighting lessening and $n$ is the number of time periods. We just utilize the EMA esteem $CW_t$ as the average CPU workload.

$$CW_1 = cw_1$$  \hspace{1cm} (3.8)

$$CW_t = \delta \times cw_t + (1 - \delta) \times CW_{t-1}$$  \hspace{1cm} (3.9)

$$\delta = \frac{2}{n + 1}$$  \hspace{1cm} (3.10)

3.6.2 BANDWIDTH PREDICTION

To predict average bandwidth (Park et. al., 2014), EMA is likewise utilized as a part of the bandwidth monitor when settling on an offloading choice. In addition, it utilizes Equations 3.11, 3.12 and 3.13 to recursively calculate EMA value $BW_t$ for bandwidth at current time $t$, where $bw_t$ is the bandwidth recorded by bandwidth monitor occasionally, coefficient $\gamma$ represents the level of weighting lessening and $n$ is the number of time periods.

$$BW_1 = bw_1$$  \hspace{1cm} (3.11)

$$BW_t = \gamma \times bw_t + (1 - \gamma) \times BW_{t-1}$$  \hspace{1cm} (3.12)

$$\gamma = \frac{2}{n + 1}$$  \hspace{1cm} (3.13)
3.6.3 ENERGY CONSUMPTION ON MOBILE DEVICE

Keeping in mind the end goal to compute the energy consumption expended by running the application on mobile device (Park et. al., 2014), we have to know the active power of the processor on mobile device \( P_e \) and the average execution time \( T_e \) anticipated by the execution time predictor, and after that we utilize Equation 3.13 to get energy consumption \( E_m \).

\[
E_m = P_e \times T_e
\]  
(3.14)

3.6.4 ENERGY CONSUMPTION ON REMOTE CLOUD

Computing energy consumption on remote cloud (Park et. al., 2014) \( E_c \) is more entangled than \( E_m \). Some time recently acquainting how to compute \( E_{cloud} \), we make a suspicion that the offloading part of the application is as of now on cloud when making an offloading choice. As mentioned previously, it needs three stages to complete computation offloading, sending the required input data, waiting the cloud finishing execution of the offloaded computation and receiving execution results from cloud. In this way, \( E_m \) cloud incorporates three sections: the energy devoured by sending required information on mobile device \( E_s \), the energy of waiting execution results on mobile device \( E_i \) and the energy devoured by receiving execution results on mobile device \( E_r \). \( E_s \) is computed by Equation 3.15, and the time for sending information from mobile device to cloud \( T_s \) is ascertained by Equation 3.16. Additionally, we ascertain \( E_r \) by utilizing Equations 3.17 and 3.18.

\[
E_s = P_s \times T_s
\]  
(3.15)

\[
T_s = D_s/B_s
\]  
(3.16)

\[
E_r = P_r \times T_r
\]  
(3.17)

\[
T_r = D_r/B_r
\]  
(3.18)

It is but difficult to get Equation 3.19, and we utilize it to represent to execution time of the application on mobile device. Since we just offload entire or part of the application to cloud, we are simple to come to Equation 3.20 and use it to present inert time on mobile device, i.e. execution time of the remote cloud-version application.

Assume that \( n \) represents the proportion of computation execution rate on remote cloud and mobile device, and afterward we get Equation 3.21. Due to generally extensive estimation of \( n \), we just utilize the most extreme value \( T_{im} \) to ascertain the idle
time on mobile device in our investigations. Along these lines, we can utilize Equations 3.22 and 3.22 to ascertain $E_i$.

$$T_e = C/M \quad (3.19)$$

$$T_i <= C/S \quad (3.20)$$

$$T_i <= T_i m \quad (3.21)$$

$$E_i = P_i \times T_i \quad (3.22)$$

Consequently, we can determine Equation 3.23 to calculate $E_c$ from Equations 3.15, 3.17 and 3.22. Besides, we get the trade-off energy consumption for computation offloading $E_{i,off}$ from Equations 3.14 and 3.23, i.e. the distinction of energy consumption of running the application on mobile device and cloud, as appeared in Equation 3.24.

$$E_c = E_s + E_i + E_r = P_s \times T_s + P_i \times T_i + P_r \times T_r \quad (3.23)$$

$$E_{i,off} = E_m - E_c = P_e \times T_e - P_s \times T_s - P_i \times T_i - P_r \times T_r \quad (3.24)$$

We change Equation 3.24 to Equations 3.25 and 3.26. At that point we make $E_{i,off}$ equivalent to zero and get a constant, called break-even transmission energy $E'_0$, as appeared in Equation 3.27. Once the application and the status of mobile device are indicated, the value of $E'_0$ is constant.

$$E_{i,off} = P_e \times T_e - P_i \times T_i - E' \quad (3.25)$$

$$E' = P_s \times T_s + P_r \times T_r \quad (3.26)$$

$$E'_0 = P_e \times T_e - P_i \times T_i \quad (3.27)$$

If $E'_0$ is more prominent than $E'$, it offloads computation of the application from mobile device to cloud. Else, it runs the application on mobile device.
3.7 VIRTUAL NETWORK COMPUTING

In computing, Virtual Network Computing (VNC) is a graphical desktop sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another PC. It transmits the keyboard and mouse events starting with one PC then onto the next, handing-off the graphical screen redesigns back in the other direction, over a network (Richardson et al., 1998).

In the VNC framework, server machines supply applications and information as well as a whole desktop environment that can be accessed from any Internet-associated machine utilizing a basic software NC. At whatever point and wherever a VNC desktop is accessed, its state and configuration (directly down to the position of the cursor) are precisely the same as when it was last accessed. The innovation fundamental VNC is a simple remote display convention. It is the simplicity of this protocol that makes VNC so powerful.

3.7.1 THE VNC PROTOCOL

The innovation underlying the VNC framework is a simple protocol for remote access to graphical UIs. It works at the framebuffer level and subsequently applies to every operating system, windowing systems, and applicationsindeed to any gadget with some type of communications connection. The protocol will work over any solid transport such as TCP/IP. The endpoint with which the client interfaces (that is, the display or potentially input gadgets) is known as the VNC client or viewer. The endpoint where changes to the framebuffer begin (that is, the windowing system and applications) is known as the VNC server.

3.7.2 INPUT

The input side of the VNC protocol depends on a standard workstation model of a keyboard and multibutton pointing gadget. The client sends input events to the server at whatever point the client presses a key or pointer button, or moves the pointing gadget. Input events can likewise be blended from other nonstandard I/O gadgets. On the Videotile, for instance, a pen-based handwriting recognition engine creates keyboard events.

3.7.3 VNC VIEWERS

In everyday utilize, we lean toward the more enlightening term viewer to the fairly over-burden word client. Composing a VNC viewer is a basic assignment, as surely it ought to be for any thinclient framework. It requires just a solid transport (generally TCP/IP),
and a method for displaying pixels (either writing specifically to the framebuffer or going through a windowing framework).

3.7.4 VNC SERVERS

Writing a VNC server is marginally harder than composing a viewer. Since the protocol is intended to make the client as simple as could be expected under the circumstances, it is for the most part up to the server to play out any important interpretations (for instance, the server must give pixel information in the arrangement the client needs). We have written servers for our two fundamental platforms, X (that is, Unix) and Windows.

3.7.5 ADVANTAGES OF VNC

In addition, VNC allows a single desktop to be accessed from several places simultaneously, thus supporting application sharing in the style of computer-supported cooperative work (CSCW). Unlike other remote display protocols such as the X Window System and Citrixs ICA, the VNC protocol is totally independent of operating system. The VNC system is freely available for download from the ORL Web site at http://www.orl.co.uk/vnc/.

3.8 ARCHITECTURE

The framework architecture is the reasonable model that characterizes the structure, behaviour, and other perspectives of a framework. A construction modelling portrayal is a formal depiction and representation of a framework, sorted out in a manner that supports thinking about the structures of the framework. The framework architecture can involve system components, the remotely obvious properties of those parts, and the connections between them.

The Software Architecture of our proposed framework is the functionality of the virtualization technology. Issues such as the energy consumption limit of mobile devices will be eliminated utilizing virtualization. The local machine may be a laptop or smart phone, and the remote machine ought to be workstation. The virtual images of physical mobile device applications are stored away in the virtual mobile device homestead, and each one picture is devoted to every customer or client.

Figure 3.3 demonstrates that the cloud is an enormous layer of assistance for new and extensive numbers of services. The act of utilizing a system of remote servers, instead of a local server or a PC, facilitates the storing, overseeing, and processing of information. In the cloud, virtual images of the mobile applications are stored away, starting in the physical mobile device. Each time the application is used on the physical mobile device, essential estimations such as calculated execution time, memory of the
application and power utilization are ascertained to make an offloading choice. If offloading is enabled, the control of the application is passed as input device events from the VNC (Virtual Network Computing) client in the local mobile device to the VNC server that is installed in the cloud. At exactly that point, the application is executed remotely, and just the output image of the application is sent to the physical mobile device as a video stream utilizing VNC. VNC is a graphical desktop sharing framework that utilizes the Remote Frame Buffer protocol (RFB) to control remotely an alternate machine. It transmits the console and mouse events starting with one machine then onto the next, transferring the graphical screen updates back in the other directions, over a system.

3.9 ALGORITHM

The steps provide in Algorithm 3.1 explains that every time the application is ac-
Algorithm 3.1 Computation Offloading for MCC

1. Predict the application’s average execution time ($T_x$).
2. Get the size of the application ($S_x$).
3. Get the delay tolerance ($T_y$).
4. Getting the available primary memory size ($S_y$).
5. Check the condition.
6. if ($T_x > T_y$) or ($S_x > S_y$) then
7. Calculate power consumption in Local $E_l$
8. Calculate power consumption in Cloud $E_c$
9. if ($E_l < E_c$) then
10. Run in local device.
11. Goto Step:53
12. else
13. Goto Step:16
14. end if
15. else
16. if (Favorable network bandwidth) then
17. if (Offloading enabled) then
18. if (Resource available) then
19. if (Offloading favorable) then
20. Run application in cloud
21. Use scalable NFS for unlimited storage
22. Use high processor available in cloud
23. Transmit keyboard and mouse events to cloud
24. Receives graphical screen update using VNC
25. Appears like running locally
26. Store the new remote session on mobile device
27. Goto Step:53
28. else
29. Establish new remote session
30. Run application in cloud
31. Use scalable NFS for unlimited storage
32. Use high processor available in cloud
33. Transmit keyboard and mouse events to cloud
34. Receives graphical screen update using VNC
35. Appears like running locally
36. Store the new remote session on mobile device
37. Goto Step:53
38. end if
39. else
40. Goto Step:10
41. end if
42. else
43. Goto Step:10
44. end if
45. else
46. Goto Step:10
47. end if
48. else
49. Goto Step:10
50. end if
51. end if
52. STOP
cessed on the physical mobile system, the basic measurements, such as the execution time ($T_x$), size of the application ($S_x$) and delay tolerance ($T_y$), are calculated on the local system to make the offloading decision. To ensure offloading, two conditions (($T_x > T_y$) and ($S_x > S_y$)) should be met. If offloading is enabled and the resources are available, the control of the application is passed as input device events from the VNC client on the local mobile device to the VNC server that is installed in the cloud. The application that is passed to the cloud utilizes scalable NFS for unlimited storage and high-speed processors. It sends the mouse and keyboard events from one machine to another, communicating the graphical screen updates send back in the other direction, through a network. Only then is the application executed remotely, and only the output image of the application is sent to the physical mobile device as a video stream using VNC. This allows applications running in the cloud to appear as local applications on the physical device, with functions such as copy-and-paste between local and remote applications. It also features remote shortcuts to remote applications in the virtual cloud storage that minimize the number of steps required for users to launch remote applications by allowing stored sessions.

3.10 EXPERIMENTAL SETUP

We assess the proposed structure by benchmarking a model application for mobile devices on the AWS (Amazon Web Services) EC2 service. The experimental setup is composed of a laptop with the processor of Intel Pentium CPU B960@2.20GHz, 4GB RAM and Windows 7 operating system, Wi-Fi wireless network, VNC and AWS.

AWS EC2 Service

Today, AWS(Amazon, 2015) provides a profoundly dependable, versatile, minimal infrastructure platform in the cloud that powers a huge number of organizations in 190 nations far and wide. With datacentres in farm areas in the U.S., Europe, Brazil, Singapore, Japan, and Australia, customers over all businesses are exploiting the accompanying advantages.

1. Low Cost
2. Agility and Instant Elasticity
3. Open and Flexible
4. Secure.

Amazon Elastic Compute Cloud (Amazon EC2) is a service that gives a resizable compute limit in the cloud. It is intended to make web-scale processing simpler for designers. Amazon EC2’s basic web service interface permits acquiring and configuring
capacity with negligible erosion. It furnishes complete control of r computing resources. Amazon EC2 lessens the time needed to obtain and boot new server cases to minutes, permitting the rapid scaling of limits as the r computing prerequisites change. Amazon EC2 changes the financial aspects of processing by permitting paying just for the limit that is really utilized. Amazon EC2 gives developers the instruments to fabricate disappointment flexible applications and disengage themselves from basic disappointment situations. The steps to interface with AWS are as follows:

1. Sign Up in AWS
2. Launch Our Instance from an AMI

The experimental setup is composed of server nodes, which run instances of the mobile virtual device, a Wi-Fi wireless network, and the VNC client on a laptop mobile device. The mobile device is employed on the server machine for the execution of the offloaded service component of the application at runtime. Mobile devices access the wireless network via the Wi-Fi wireless network connection, with the available physical layer data rates. Battery care software is used for the measurement of battery power consumption on the mobile device in distributed application processing.

3.11 RESULTS AND DISCUSSION

3.11.1 COMPARING POWER CONSUMPTION OF THE LOCAL SYSTEM WITH THAT OF THE CLOUD SYSTEM

To calculate the power consumption, battery care software is used. Using the current capacity feature given in the battery care software, the difference between the current capacities of the system before the application starts running and that after the application finishes running is determined.

3.11.2 CALCULATING POWER CONSUMPTION IN THE LOCAL SYSTEM

We had to make a note of the current capacity of our system before executing the application. Only then should the application be executed. After finishing execution, again note the current capacity of the system. One important thing we need to keep in mind is that we have to close or suspend all other background programs running in the system. Finally, calculate the difference between the two capacities to find the total power consumption.

3.11.3 CALCULATING POWER CONSUMPTION IN THE CLOUD SYSTEM

Firstly, we need to suspend all other background programs running in the system. Using the IP address and password created for our remote instance, the login is validated and
control is passed to the cloud. We have to make a note of the current capacity of our system before executing the application. Only then should the application be executed. In the cloud system, select the application to execute. After finishing execution, again note the current capacity of the system. Finally, calculate the difference between the two capacities to find the total power consumption.

3.11.4 CALCULATING THE POWER CONSUMPTION DIFFERENCE

Find the difference between the power consumption calculated for the local system and that of the cloud system.

3.11.5 CALCULATING THE ENERGY CONSUMPTION OF THE PROPOSED OFFLOADING ALGORITHM WITH TRADITIONAL OFFLOADING ALGORITHM

The evaluation is done by setting up a similar mobile and cloud environment for offloading applications between them. The operational costs are compared taking into account the factors like speed, energy consumption and efficiency of the video offloading. The result shows the prominence of the proposed framework over other related ones considering their theoretical offloading strategies available in literature.

3.11.6 THE CURRENT CAPACITY OF THE BATTERY AFTER ONE HOUR

CALCULATION OF THE POWER CONSUMPTION TO PLAY A VIDEO FOR ONE HOUR IN THE LOCAL MACHINE

We calculate the energy consumed by the video application with the size of 703MB of .mkv file execution on the local machine without using our framework. The battery capacity when starting the video was 32378 mWh as shown in Figure 3.4. One hour of forced video play was conducted, and, at the end of it, the capacity of the battery was 21244 mWh as shown in Figure 3.5. By that, we can conclude that the energy consumed by playing the video for one hour was 11134 mWh.

CALCULATION OF THE POWER CONSUMPTION FROM PLAYING A VIDEO FOR ONE HOUR IN THE REMOTE MACHINE

To prove that the proposed framework is energy efficient, a simulation is performed using a setup. The simulation is conducted using videos of different quality and byte size. The framework response to the video processing over the cloud was noted. The battery capacity when starting the video was 32033 mWh as shown in Figure 3.6. One hour of forced video playback was conducted, and, at the end of it, the capacity of the battery
Figure 3.4 Local execution of a video application and the current capacity of the battery at the start of the video execution

Table 3.1 Energy consumption of local execution and remote cloud execution and the percentage of energy saving

<table>
<thead>
<tr>
<th>$T_{ex}$ (in hrs)</th>
<th>$E_{ext}$ (in mWh)</th>
<th>$E_{ecc}$ (in mWh)</th>
<th>% of energy saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>8333</td>
<td>3131</td>
<td>62.00466</td>
</tr>
<tr>
<td>1.0</td>
<td>11134</td>
<td>3705</td>
<td>66.3565</td>
</tr>
<tr>
<td>1.5</td>
<td>12600</td>
<td>3849</td>
<td>69.11614</td>
</tr>
<tr>
<td>2</td>
<td>14841</td>
<td>4214</td>
<td>71.29145</td>
</tr>
</tbody>
</table>

$T_{ex}$: Time consumed to execute the video application (in hrs);
$E_{ext}$: Energy consumption from local execution (in mWh);
$E_{ecc}$: Energy consumption from remote cloud execution (in mWh)

was 28328 mWh as shown in Figure 3.7. By that, we can conclude that the power consumed by playing the video for one hour was 3705 mWh. The proposed framework is compared with traditional offloading frameworks for execution time, energy consumption and computational cost. The results obtained after the different chosen applications are run by offloading them to the cloud using the proposed and traditional offloading algorithms shows the efficiency of the proposed algorithm over the related ones for a same contextual scenario’s.

Figure 3.8 shows a comparison of the energy consumption of the video application execution on the remote cloud and the local machine. The energy efficient computational offloading framework shows reduction in the energy consumption of 66.36% for the duration of one hour. The various durations and energy consumptions on the local machine and remote cloud, as well as the percentages of energy savings, are shown in Table 3.1.

Figure 3.9 shows a comparison of the energy consumption of the video application execution on the local machine and via the computational offloading system for various durations. It shows that the energy consumption from use of the computational
Figure 3.5 Local execution of a video application and the current capacity of the battery after one hour of video execution

offloading system decreases compared to local execution.

**CALCULATION OF THE POWER CONSUMPTION TO GENERATE A RANDOM NUMBER APPLICATION IN THE LOCAL MACHINE**

We calculate the energy consumed during the random number generation application execution on the local machine using traditional framework. The battery capacity when starting the execution of the application was 31628 mWh as shown in Figure 3.10 and ending of the application was 30348 mWh as shown in Figure 3.11. By that, we can conclude that the energy consumed to execute the application was 1280 mWh.

**CALCULATION OF THE POWER CONSUMPTION TO GENERATE A RANDOM NUMBER APPLICATION IN THE REMOTE MACHINE**

The battery capacity when starting the execution of random number generation application was 31028 mWh as shown in Figure 3.12, at the end of the execution the capacity of the battery was 30348 mWh as shown in Figure 3.13. By that, we can conclude that the power consumed to execute the application was 680 mWh. The proposed framework is compared with traditional offloading frameworks for execution time, energy consumption and computational cost. The results obtained after the different chosen applications are run by offloading them to the cloud using the proposed and traditional offloading algorithms shows the efficiency of the proposed algorithm over the related ones for a same contextual scenario’s.

In traditional computational offloading, the average energy consumption of the sorting techniques is reduced by 36% for the data size of 40,000, and the average energy
Figure 3.6 Remote cloud execution of the video application and the current capacity of the battery at the start of the video execution.

Figure 3.7 Remote cloud execution of the video application and the current capacity of the battery after one hour of video execution

The consumption of the matrix multiplication is reduced by 10.9% for the size of $440 \times 440$. However, with the Energy Efficient Computational offloading Framework, the average energy consumption of the video application is reduced by 67.19%, and random number generation application is reduced by 46.87% which is shown in Figure 3.14 as well as the concerned numerical measures are shown in Table 3.2.

### 3.12 SUMMARY

These results demonstrate that the power consumption can be reduced through offloading because complex applications are offloaded to the remote cloud. Much energy can be saved, which indicates that users can have more battery time compared to using local execution. The results also prove that the overhead of our proposed framework is
small. Our proposed framework supports the offloading of mobile computing devices to remote clouds. It was found that an average energy savings of 67.19% is achieved while executing a video application over a period of two hours and random number generation application is reduced by 46.87% compared to the traditional method.
Figure 3.11 Local execution of a random number generation application and the current capacity of the battery after execution.

Figure 3.12 Remote cloud execution of the random number generation application and the current capacity of the battery at the start of the execution.

Figure 3.13 Remote cloud execution of the random number generation application and the current capacity of the battery after execution.

Figure 3.14 Comparison of the energy savings with existing applications.
### Table 3.2 Comparison of the energy savings with existing applications

<table>
<thead>
<tr>
<th>Applications</th>
<th>Average % of energy savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existing application</td>
<td></td>
</tr>
<tr>
<td>Sorting</td>
<td>36</td>
</tr>
<tr>
<td>Matrix multiplication</td>
<td>10.9</td>
</tr>
<tr>
<td>Tested application</td>
<td></td>
</tr>
<tr>
<td>Random number generation</td>
<td>46.87</td>
</tr>
<tr>
<td>Video application</td>
<td>67.19</td>
</tr>
</tbody>
</table>