Computational Analytics of Client Awareness for Mobile Application Offloading with Cloud Migration

Uma Nandhini* and Latha TamilSelvan
Department of Information Technology, School of Computer, Information and Mathematical Science
B.S.Abdur Rahman University, Vandalur, Chennai-600048, Tamil Nadu, India
[e-mail: umaudhay@gmail.com, latha.tamil@bsauniv.ac.in]
*Corresponding author: Uma Nandhini

Received May 14, 2014; revised July 26, 2014; revised September 3, 2014; accepted September 10, 2014; published November 30, 2014

Abstract

Smartphone applications like games, image processing, e-commerce and social networking are gaining exponential growth, with the ubiquity of cellular services. This demands increased computational power and storage from mobile devices with a sufficiently high bandwidth for mobile internet service. But mobile nodes are highly constrained in the processing and storage, along with the battery power, which further restrains their dependability. Adopting the unlimited storage and computing power offered by cloud servers, it is possible to overcome and turn these issues into a favorable opportunity for the growth of mobile cloud computing. As the mobile internet data traffic is predicted to grow at the rate of around 65 percent yearly, even advanced services like 3G and 4G for mobile communication will fail to accommodate such exponential growth of data. On the other hand, developers extend popular applications with high end graphics leading to smart phones, manufactured with multicore processors and graphics processing units making them unaffordable. Therefore, to address the need of resource constrained mobile nodes and bandwidth constrained cellular networks, the computations can be migrated to resourceful servers connected to cloud. The server now acts as a bridge that should enable the participating mobile nodes to offload their computations through Wi-Fi directly to the virtualized server. Our proposed model enables an on-demand service offloading with a decision support system that identifies the capabilities of the client's hardware and software resources in judging the requirements for offloading. Further, the node's location, context and security capabilities are estimated to facilitate adaptive migration.

Keywords: Mobile cloud computing, computation offloading, cloud migration, Wi-Fi, client aware cloudlet, power aware computing

http://dx.doi.org/10.3837/tiis.2014.11.014

1. Introduction

Technological advancements in wireless communication and mobile technology in portable devices have become integrated into the fabric of everyday life. With increased mobility, standalone applications and remote access mobile applications have become the prime focus of the next generation app developers and their users. This trend of the mobile ecosystem is fast colluding with the services offered by cloud computing, to form the newly emerging mobile computing paradigm called Mobile Cloud Computing. This shift in Information Technology and on-demand service delivery driven by economies, paved the way for this transition and paradigm shift towards next generation computing. Cloud computing provides a shared pool of virtualized resources that are dynamically configurable and managed by computing resources, which are delivered on demand to the end user over the available networks. It provides flexibility and mobility of information, by abstracting and decoupling it from the underlying technology, and offers it at a negligibly low cost of the pay per use model. However, the actual realization of these benefits is far from being achieved for mobile applications. According to Gartner, mobile phones are expected to overtake personal computers as the most common web access devices, which will be an entry point and interface for cloud online services. Smartphones, being more superior to basic mobile phones, possessing higher computation and sensors monitoring various parameters with advanced connectivity, are used for video calling, social networking, e-commerce, gaming, GPS navigation etc. In spite of all these advancements, smart phones are always resource-constrained when compared to their non-mobile counterparts, because smart phones are battery powered and dissipate heat that affects their computational capabilities. Moreover, the processors that are in use are primitive in technology and their storage is absolutely limited to their size. Even their wireless connectivity has bounds with interference, when compared to wired networks. Therefore, mobile phones are incapable of executing high end offline applications like augmented reality, image processing, file format conversions, face recognitions, business applications and other multimedia rich applications. On the other hand, connecting to online applications directly, using mobile internet via 2G or 3G requires an optimal data rate with no network congestion. Going by the present scenario, the mobile-connected devices will soon exceed the human population on earth, and with 10 billion devices will create an inevitable '*digital data storm*', if connected. The growing demand for rich multimedia content and applications online needs better data capacity of mobile operators, which in turn, leads to increased capital and operational expenditure. Increases in cost of the order of tens of millions on new infrastructure will have to be borne by the subscriber, leading to an increase in the average billing cost. Hence, a location based infrastructure like Wi-Fi networks will alleviate the congestion problem and network availability to combat the '*digital traffic jam*'. To achieve adaptable computation facilities, either hardware or software, needs a capable technological transformation. Hardware changes can be done to a certain extent, and they are manufacturer specific, where the battery performance grows only at 5 percent when compared to the exponential growth of other technologies. The only possible solution is at the software level, by provisioning the computations to nearby resourceful servers.

Computation offloading is a procedure, whereby mobile devices identify any nearby adaptable infrastructure that has the potential to compute the given task at a substantially reduced time, and send the results back to mobile devices. This process can drastically lessen the battery power consumption, and increase the related hardware performance. This is different from the traditional client-server architecture, where a thin client always migrates computation to a server. Computation offloading is different from the migration model used in multiprocessor systems and grid computing, where a process is migrated for load balancing [1]. The purpose of this paper is to provide a vision for the future, whereby these mobile devices can exchange data that are geographically constrained. Some of the enabling applications that could benefit from offloading data to the nearby servers are, for example, a robot that navigates around and needs to identify every object in its path. If the processor is slow, then a real time analysis based on location awareness could not be done. Another application that ascertains context awareness is, multiple sensors connected to a user, sending real time information like GPS, temperature and humidity which need analysis; then offloading proves to be an effective solution. For a client aware computing, consider multiple clients that participate in an ad-hoc network; if requirements like power and processing capabilities are a constraint for the client, the client could be supported by the surrogate system.

The objective of our work is to propose a cloudlet framework in a mobile cloud environment, where it provides a transparent elastic augmentation of mobile device capabilities via ubiquitous wireless access, to cloud storage and computing resources. Also, our paper focuses on the context and client awareness for dynamic adjusting of offloading to changing operating conditions, by preserving the available sensing and interactivity capabilities of mobile devices. The following sections discuss the related works and issues involved. In section 3, we provide detail system architecture of the offloading computation model, and section 4 presents an algorithm for offloading conditions and a decision support system. In the last section an experimental analysis follows after the implementation scenario.

2. Related work

Recent Literatures on offloading computations, due to inherent drawbacks of cellular services and smart phones, have clearly determined that considerable research is focused towards future generation mobile cloud computing. One of the important works for offloading data to the nearby server is from Satyanaranyana [2], proposing a model for cloudlet based resource rich mobile computing, where there is a real time interactive response with low latency, one-hop, high-bandwidth wireless access to the resource rich server, called cloudlet. It highlights the proximity of mobile devices to the cloudlet, and if it is connected through wireless, then VM can be synthesized in the cloudlet. The paper proposes these concepts as a vision, and there are multiple deployment challenges to be met before it is widely accepted.

Bo Han et al. [3] propose an opportunistic communication to facilitate information dissemination in the emerging Mobile Social Networks (MoSoNets), thereby reducing the amount of mobile data traffic. Offloading data between mobiles in Wireless LAN Hotspots uses greedy heuristics, and random selection algorithms, to identify target users and target application. Application-oriented offloading of computation is carried out in [4] where an open, extensible architecture to enable context-aware navigation for the blind and visually impaired, using a collaboration model of mobile and location specific information resource is provided. High quality navigation guidance with rich context awareness using cloud maps, is modeled. A potential application scenario that could be used for offloading data is suggested by Nguyen [5], where a hybrid positioning system uses the mobile phone sensors for indoor navigation. A dynamic offloading algorithm, using Lyapunov optimization, is proposed by

Dong Huang et al. [6]. The algorithm has low complexity to solve the problems, of which software component needs to be offloaded, under the given network connectivity. The mathematical model is simulated and the experimental results show that a considerable amount of battery can be saved, by ensuring that the application execution time satisfies the given time constraint.

Other researches provide a generic survey that highlights the importance of data offloading through the concept of surrogate systems and cyber foraging, and further gives an understanding of how mobile cloud computing functions. Marinelli [7] proposed an application model called Hyrax, where the data is offloaded to the cloud, and the processing is done, using map-reduce concepts of Hadoop framework. A learning and research environment for computational science on mobile devices, to utilize the supercomputing software like the compressible flow solver and Nano device simulation tool, was developed by Park et al. [8]. An intelligent access scheme is emphasized by authors Klien et al. [9] and Dinh et al. [10] for an understanding of various architectures, applications and approaches. In [11], the authors present basic level comparison of the application models. However, as location based applications are more prevalent in mobile phones, Lian Wang et al. discuss the importance of saving energy when GPS based mechanisms are used [12]. The paper also discusses how location updates and trajectory data can be simplified. Further, it provides a survey of how other tracking applications perform when compared to their dynamic approach. A novel approach developed by Fuhong [13] suggests a pre-pushing and downloading model in a mobile peer-assisted streaming network to perform resource caching for a demanding service. The optimal speed of downloading has been analysed using Bellman's theory to achieve the Nash Equilibrium. Having surveyed these papers, it can be understood that the cloud is fast getting integrated into smartphones, to provide an excellent amalgamation of service delivery to overcome the challenges and constraints faced by the users.

2.1 Issues and challenges

Issues pertaining to offloading involve the efficient utilization of the nearby resources through timely provisioning and scheduling. The challenges in designing an efficient migration mechanism in a wireless device are cloudlet and cloud servers, because the workload may not be stable due to unknown number of participating nodes. Some of the issues that need to be addressed, in order to make an offloading decision are as follows.

Interoperability involves situations where several nodes connect to different types of networks, and possibly the switching between networks does not affect the desired functionality of the mobile devices. For example, a navigation robotic system might be connected with all the generations of cellular networks along with Wi-Fi, Bluetooth, infrared and NFC (Near Field Communication). During the offloading process, if only the Wi-Fi is currently made available, then the other sensors could not execute their desired functionality, which leads to the failure of the entire system. Hence, interoperable communication capabilities that hide this interaction are an important design issue.

Context awareness perceives the user's state and surroundings, based on which it determines the application's behavior. So, offloading should be done dynamically, by effectively analyzing the present context, and adapting the system to different situational constraints. It may be an active context where the application automatically adapts to the discovered context, by changing the application's behavior, or a passive context awareness in which an application presents a new or updated context, and makes it possible for the user to retrieve it later.

Location awareness identifies the device location using a navigational system like the GPS (Global Positioning System) or WPS (Wireless Positioning System), and making an effective decision in predicting the movement. Issues related to routing based on location are key factors to deliver the voluminous data in the shortest route, in case the node is out of reach for the surrogate system, but still connected through the wireless network.

Client awareness judges the capabilities of the client devices, using a central profile management, to provide flexibility to meet the expanding user needs. Device attributes differ in terms of screen size, input method, processing power and available memory; hence, the cloudlet should fine-tune the services depending on these attributes. For example, a device can perform more when used in Wi-Fi than with other networks, or it can have advanced capabilities for rendering images.

Security and Privacy is a prominent issue when mobile devices are integrated with cloud [16], [17]. In general, mobile devices are vulnerable to viruses, worms and other malicious codes, due to the infeasibility of their performing a complete security scan, while providing various services. Hence, an untrustworthy adversary may inflict damage by injecting malicious codes that may change an application's behavior, thereby compromising the integrity of the user's data. Security issues like data corruption, data leakage, and denial of service, and privacy issues like identity theft, data privacy and location monitoring, are the challenges faced during mobile cloud computing. These issues could be addressed by implementing secure systems through hardware based encryption, and certificate based cryptographic encryption, thus providing a trusted environment [14], [18].

Resource allocation and Scheduling is required when multiple nodes participate in the offloading process [22], [23]. When the workload is heavier or when there is a dynamic change in the wireless environment due to mobility, then proper provisioning of the resources to the available dedicated servers requires a scheduling algorithm [5]. Also, in case of a fault, the network must be tolerant enough to allocate resources temporarily to another server, until the fault is rectified.

Power awareness is the most important issue in any wireless device, and in the present scenario it is the most significant justification for offloading, leading to the evolution of mobile cloud computing. Analyzing the battery power saved, will make the decision support system in the mobile to judge the conditions for offloading. Our work focuses on the issues of client awareness and power saving attributes, by offloading a video file for conversion.

3. Computation Offloading

Offloading Computation to a nearby server for common utility applications is the primary objective, and then migrating it to a cloud for complex applications, is the foundation of our research. So, the need for a nearby Resource-rich Middleware (RM) server in the initial phase and sidelining direct connectivity to the cloud results in huge cost benefits. For a smartphone to be connected to the cloud, it should always be connected to the mobile internet with subscription, depending on data traffic. Since most of the applications are complex, the data rate will be heavy, and this can lead to huge cost and traffic, and subsequently network congestion. Apart from this, every cloud service offered has to be paid depending on the time and usage.

$$\text{Cost of Cloud offloading} = \text{Mobile Internet cost} + \text{Cloud Service Cost} \tag{1}$$

$$\text{Mobile Internet Cost} = \text{Cost per Byte Transferred} * \text{Bytes Offloaded per Application} + \text{Cost of Power Utilized} \quad (2)$$

$$\text{Cloud Service Cost} = \text{Cost of Bytes Transferred} + \text{Cost of Service Type} \quad (3)$$

This is just the normal usage cost for every consumer, who uses cloud services through the internet. Other important hidden costs include the battery, whose life depreciates when too much of computation keeps the mobile running all the time, and the heat so generated from the battery may affect other components as well. Thus offloading computation to a nearby resource rich server via Wi-Fi is a one hop solution involving high data rate connectivity with less energy consumption. Connecting to a nearby server through Wi-Fi is extremely simple using a Wi-Fi router acting as an access point. The configuration of a resource rich server for the offloading process need not necessarily have to be a multi-processor blade server or rack server, because for computing mobile based applications, a desktop or laptop computer with dual core processing and sufficient memory, is enough. Nowadays, most of the users spend more time on their smartphones than on their desktops; these idle machines could be made as one of the offload processing machine or surrogate system. More than that, the growing popularity of the Wi-Fi hotspot around the globe is offered at a negligible cost everywhere and anyone can create a hotspot in their vicinity in seconds. The vision of this computation offloading leading to mobile cloud computing, will make the Wi-Fi hotspots that are now limited to hotels, cafes, airports, etc., to be extended to residential localities, apartment complexes, hospitals, enterprise buildings, educational institutions, and event locations. Thus, the next generation mobile communication system could well be the scenario of Wi-Fi Service management platform.

Our aim of offloading mobile data to a Wi-Fi connected system, involves several key infrastructures like the Wi-Fi Access Points (AP), Computational Analytics Engine (CAE), Mobile User Interface Application (MUIA), Client Awareness and Assessment Protocol (CAAP), Cloud Migration Service (CMS) and links to multiple Cloud Services. The architecture of our proposed model is depicted in **Fig. 1;** it shows how promptly the system work in coordination, to assess the need for an application and when to migrate the service.

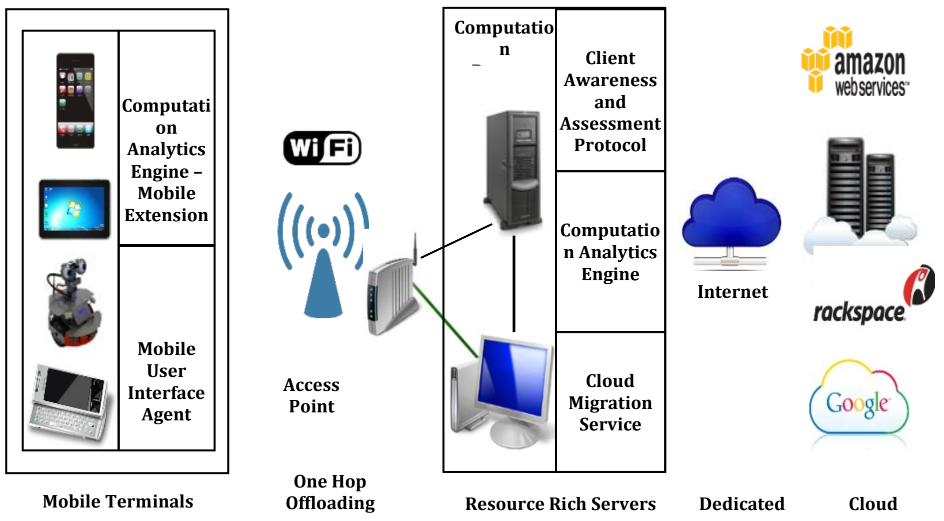

Fig. 1. Architecture of the Computation Offloading Process

There are two stages in our architecture: offloading computation to a cloud-like offline resource rich system, and online cloud migration. The primary focus of our research is only in the offline mode, and to satisfy certain requirements we move on with migration to the cloud.

3.1 Stage 1: Offloading to Resource Rich Offline Servers.

In this stage, the communication is done via a wireless interface, preferably a Wi-Fi device, connecting the two end-points, namely, the user's mobile devices/terminals and the offline resource rich server. The mobile interface is a client program that accepts the use case and identifies the type of application, and performs a basic check to locate the access point. The interface acts as an agent between the Computational Analytics Engine of the mobile terminal and the offloading server. Most of the analysis for taking a decision to offload is based on several criteria, which are undertaken at this end. Once the application is launched for service, the decision criteria that are needed to be taken and analysed by the CAE, are:

- Perform an authentication to establish the identity of the mobile user.
- Perform the handshaking process to exchange protocols for data communication.
- Whether the mobile processor can process the application.
- If it can, then how much time will it take to complete the task?
- Power utilized if the computation is done within the mobile terminals.
- Calculate the time taken for the application to synchronize and compute at the server.
- Time taken to transfer and receive the application code for the given bandwidth.
- Compare and take a decision whether to offload or compute within.

Once the application is offloaded as per the decision made at the user's end, then for better service the surrogate's CAE must:

- Identify its clients and analyze the attributes of the clients
- Identify the application and its attributes
- Identify the context and behavior
- Ensure security capabilities of client devices
- Location based service offloading

Now, the application is offloaded, and the computation process begins in the surrogate server. Meanwhile, the CAAP judges the ubiquitous nature of the mobile device, and provides an alternative to back up the completed application in the cloud storage. This would be done, only if the mobile is out of range, or the MUIA requested an option in its initial settings.

3.2 Stage 2: Offloading to Cloud Servers

In this stage, the offloading for computation can be done either directly to the cloud servers, or indirectly by the CAAP, based on the resource availability of the surrogate system. Direct offloading to cloud will be defined by the initial user preference that is provided as an additional component in the MUIA. Such a kind of situation arises when:

- The user might be moving away from the vicinity of the surrogate system as soon as he offloads the complex task.
- The device's battery is limited, which could enable to offload but could not receive.
- When the user thinks that the device would be busy with another task that has a higher priority, and is not in a position to get interrupted.

Indirect offloading is done with the judgment made by the client awareness, and the assessment protocol and computational analytics engine. Some of the scenarios for indirect offloading are:

- If the given application could not be found in the intermediate server, this is because the surrogate system would have applications only for a frequently used one.
- Complex applications like map-reduce, recommendation services engine, and non-compatible applications would be found only in online cloud servers.
- Client awareness and context awareness may force the system to forward directly to be serviced by cloud providers.
- Some applications may be only cloud specific, because of the device's direct connectivity with a paid cloud service account.

4. Proposed offloading decision algorithms

The decision to offload involves multiple situations, which are needed to be monitored before and after the process of offloading the task for computation. Before any task is offloaded, the system must analyse the requirements for the process, and set a threshold for initiation. After some basic steps are completed, the task is set for offloading to the intermediate server for computation; while doing so, the other parameters like distance, client capabilities and their context are computed. The most important criterion which is also the foremost issue in the formation of the mobile cloud, is client aware computing, which includes power management and process management. The following subsections define the algorithm for various parameters that contribute to the success of the computation offloading process.

4.1 Client Awareness and Assessment Protocol

The cloud Infrastructure must accommodate wide ranges of client devices, from smartphones, tablets and digital assistance to other automated embedded devices that are connected to the cloud. The performance and security issues that are manageable by these clients are very limited in our computing environment. Hence, development towards intelligent clients that have advanced management infrastructure with enhanced energy aware performance features is a key to sustain the growth of mobile devices. One such enhancement to the existing mobile infrastructure is the introduction of static computers to support the resource constrained mobile devices through resource augmentation. While introducing the computers as a surrogate system we need to assess multiple clients' computing power along with their capabilities and context. For delivering quality user experience, the computational capabilities of the mobile devices must be assessed before taking a decision to offload the task. Once the decision to offload has been taken, then the attributes are stored in the server for successive computational policy analysis. Some of the clients' attributes are listed in **Table 1**. These attributes help the resource-rich system to identify the resource-poor devices, and keep the information in store for future reference. The protocol that defines the CAAP can be sequenced in the following steps, based on the sequences diagram and flow diagram, as explained in the subsequent section using **Fig. 2** and **Fig. 3** respectively.

Table 1. Client Attributes

Device Attributes	**Application Attributes**	**Context Attributes**
Mobile Device USSID	Type of Application- Offline or Online	Battery Power
MAC address	Size of Application	Type of Service
Network Capability-Bluetooth/Wi-Fi	Memory Utilized when running	Social Situation
Security Certificates	No of Threads per Application	Time Context
Privacy Constraints	No of times offloaded	Geographical Location
Hardware & Memory Configurations	Other applications used through Wi-Fi	Mobility pattern
Operating System	Concurrent Processes handled by the Operating System	Context Aware Scheduling

Step 1: The decision to offload starts when a load imbalance is triggered, due to the initiation of an application for the first time.

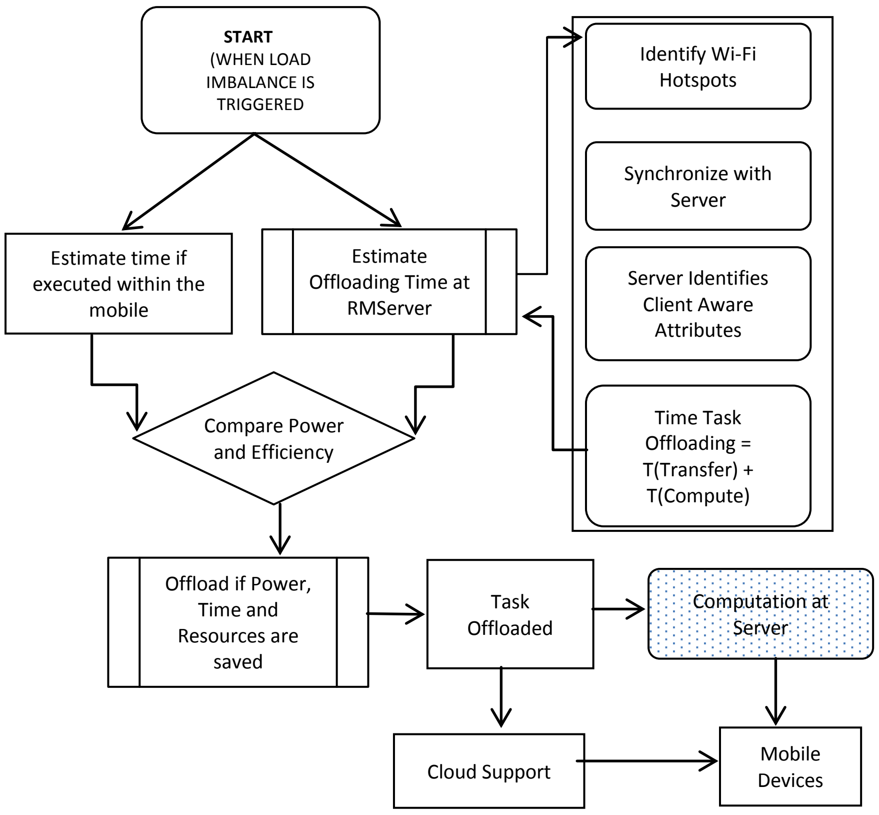
START (WHEN LOAD IMBALANCE IS TRIGGERED
Estimate time if executed within the mobile
Estimate Offloading Time at RMServer
Identify Wi-Fi Hotspots
Synchronize with Server
Server Identifies Client Aware Attributes
Time Task Offloading = T(Transfer) + T(Compute)
Compare Power and Efficiency
Offload if Power, Time and Resources are saved
Task Offloaded
Computation at Server
Cloud Support
Mobile Devices

Fig. 2. Offloading Decision

```
1.  Mobile Process Threshold {
2.          Identify ApplicationType, ApplicationSize
3.          For Task   > Threshold {
4.                  Trigger LoadImbalance, Start MUIA                {
5.                  Register DeviceUSSID, DeviceMACID with RMServer   }
6.                                  Initiate CAEmobileextension
7.                  Proceed to OffloadingDecision
```

Step 2: The decision to offload depends on the estimated time to execute the process, or if the task of the given application in the mobile is larger than the time taken by the RM Server. This can be estimated by emulating the application to execute a part of the task. Every running task comprises of millions of instructions, and a part of it could be initialized to estimate the total number of instructions. Given the processor's execution speed and instruction size, the time is calculated.

```
8.  Execution Time inMobile{
9.  InitializeApplication
10. Identify TotalInstructions for Task, ProcessorSpeed
11. Time= TotalInstruction/Processor Speed                      }
```

Step 3: Once the time taken for mobile computation is estimated, then, on the other hand, the same task is sent over the Wi-Fi medium to the server. At the server, the registered mobile is validated, and the same process of time calculation is done. The total time for execution and transfer is summed, for comparison with the execution time in the mobile.

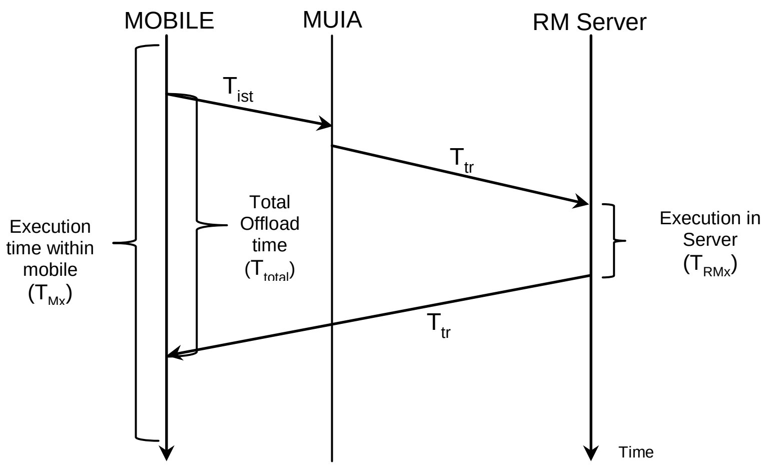
MOBILE
MUIA
RM Server
T_{ist}
T_{tr}
Execution time within mobile (T_{Mx})
Total Offload time (T_{total})
Execution in Server (T_{RMx})
T_{tr}
Time

Fig. 3. Offload Protocol Time Estimate

```
12. OffloadDecision
13. if{
14. ExectionTimeinMobile ($T_{Mx}$) >> TotalExecutionTimeinServer ($T_{total}$)
15. PowerConsumedinMobile >>PowerConsumedinMobilewhenOffloaded }
16. else { ExecutewithinMobile
```

Step 4: Conditions for the threshold are set to offload the task to the nearby server, if the power and time are saved considerably. The threshold condition is a variable that depends on the application that is being executed.

```
17.  ExecutionatServer{
18.  if
19.     TaskComplete { Send to MobileDevice     }
20.  else   (Identify Context and Location of Mobile )
21.       Notify = TimeToLive of file in RM Server
22.       NotificationTime =0;
23.  then
24.       Store in CloudStorageServer for later access     }
```

Generally, when an application, like a full length movie of size greater than 500MB, is to be converted to another format, then the estimated time and power would prefer offloading as the solution for faster execution. To accelerate the performance of the application being executed by the server, a distributed computing approach could well be the solution for enhancement, by having more offloading servers computing the task simultaneously. Once the execution is completed, the task is sent back to the mobile device if available nearby; else, the completed file is stored in the cloud server for future access. The decision to push the task to the cloud storage is taken, by assessing the CAAP and the initial status of the client's context and present status of location. In situations, where the client is too far away, then a notification is sent to the client for immediate access of the file that is temporarily being stored in the RM server. A Time to Live (TTL) for the file in the RM server, signals the client's deadline to access the file. This notification may avoid pushing the file to the cloud, thereby saving the storage cost in remote third party cloud servers.

4.2. Power Aware Computation

Of the many surveys undertaken worldwide for smartphone users, energy efficiency and power are the main features that they encounter as a problem in mobile devices. Currently, smartphones are used not only for voice-based communication, but for multiple applications of day-to-day life, that acquire information on the fly. Thus, energy is a primary constraint of mobile devices that need to be addressed through a power aware computing model [15]. The following analysis explains the condition when offloading is required, and how to improve performance through formulation.

Energy consumption is directly proportional to the number of instructions I to be executed by the processor locally I_m, where I_m ranges from $\{0 - \alpha\}$ *and* T_m is the instruction execution time for I_m instructions. M_x is the execution speed of the mobile and P_m is the power utilized at the local execution that is within the mobile.
The energy utilized/consumed at the mobile terminal is given in equation (4).

$$E_m = \frac{P_m * \ I_m * T_m}{M_x} \tag{4}$$

The Computation of N instructions in a mobile takes N/M_x seconds where N>0 and $Mx \geq 0$. Similarly, the computation of N instructions in the server takes N/S_x seconds, where S_x is the execution speed for the task in a resource rich server in other words it is the processor speed of RM Server. The energy E_s, utilized at the offloading server is

$$E_s = \frac{P_s * I_s * T_s}{S_x} \tag{5}$$

Eq. 5 depicts only the energy consumed by the server, but to compute the complete energy requirements, we need to identify the other parameters. Some of the external and internal parameters are transfer energy, idle energy, additional coding transfer, internet support and initial setup time.

Let D be the task or data that is to be offloaded, and hence, $D \in I$, where D's size could be limited to a threshold value D^t_s. The threshold value is probably a variable that depends on the type of application. Consider for example, in a file conversion the threshold depends on the bandwidth (B_{wifi}), client device capabilities (M_{proc}) and file size (D_{size}).

$$D \in I \{D >= D^t_s\} \tag{6}$$

$$\text{where, } D^t_s \text{ belongs } B_{wifi} \wedge M_{proc} \wedge D_{size} \tag{7}$$

It can be inferred from the above equation, that for an ideal condition to improve the performance of offloading, first the amount of data exchanged should be minimum: second, the bandwidth of Wi-Fi should be in a range with good signal strength, and third, the surrogate system should reasonably be with a server configuration.

Since the data transfer is connection oriented through Wi-Fi, the available bandwidth B_{wifi} is a dependent variable for offloading, and the total time T_{total} to execute the task is given by eq.(8).

$$T_{total} = T_{tr} + T_x \tag{8}$$

Here T_{tr} is the transfer time of the data in the Wi-Fi medium without accessing any internet service. It includes the time to transfer the raw data from mobile to RM server, and time to transfer the completed data from RM server to mobile client. T_x is the time taken for the RM server to execute N number of instructions, which is given as the Total number of instruction/Processor Speed of the server, as mentioned below in eq. 9.

$$T_{total} = \frac{D}{B_{wifi}} + \frac{N}{S_x} \quad ; \quad where \begin{cases} T_{tr} = \frac{D}{B_{wifi}} \\ T_x = \frac{N}{S_x} \end{cases} \tag{9}$$

But, the mobile needs some time to initiate the request; so

$$T_{total} = T_{tr} + T_x + T_{ist} \tag{10}$$

where T_{ist} is the Initial Set up Time. Now, if the computation process requires codes also to be offloaded from the mobile, then the time taken for the code T_{code} to be sent to the server needs to be considered. Also, if the code is not available in the mobile, and if it needs to be downloaded from the cloud, then the extra time T_{cloud} for cloud service should be included in the total computation time.

$$T_{total} = \begin{cases} T_{tr} + T_x + T_{ist} + T_{code}, & offloaded\ from\ mobile \\ T_{tr} + T_x + T_{ist} + T_{cloud}, & services\ from\ the\ cloud \end{cases} \tag{11}$$

Thus, the total energy to offload includes the transfer power, power to compute by the server, power used during initiation by the mobile, and the idle power of the mobile while the server computes.

$$E_{Offload\_total} = P_{tr} * T_{tr} + P_s * \frac{I_s}{S_x} + P_{ist} * T_{ist} + P_{idle} * T_{idle} \tag{12}$$

Thus, in order to offload, the energy utilized or consumed by the mobile should not be greater than that for not offloading, i.e.

$$E_m - E_{offload\_total} >> 0 \tag{13}$$

4.3. Process Aware Energy Computation

When not offloading, i.e., when the task is computed in the mobile terminal itself, then we can also consider the energy consumed by other applications and processes that are running concurrently.

$$E_m = K * \left(\frac{P_m * I_m * T_m}{M_x}\right) + \left(\frac{P_m * I_m * T_m}{M_x}\right) \tag{14}$$

where K is the number of other applications running concurrently.

4.4. Cloud Migration for Computation

The option of cloud migration for computation offloading arises, when the context and capabilities of the resource-rich server are constrained due to various factors. Consider a situation of our earlier problem of a video file format conversion. In this case, the target server should indeed have the necessary codecs for the requested format. If it is not found in the RM Server then, the user need not be denied the option of offloading, but can be given an option of migration to cloud services. Migration of computation to the cloud enhances the idea of mobile cloud computing to the next level boundary less mobile access.

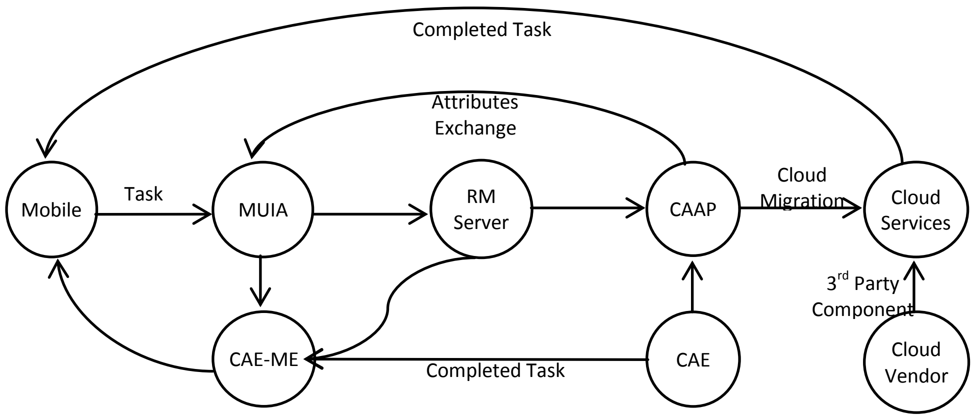

Fig. 4. Data Flow for Cloud Migration

Having ported the files to the cloud server, the cloud services will identify the task and execute accordingly. The execution of the video file format conversion will be done at a cost that includes the cost of storage and third party component, which is necessary for codecs integration. The cloud migration and its services are initiated as in **Fig. 4**, by the attributes' exchange mechanism, between the server's client aware protocol and the user interface.

5. Implementation

We have evaluated our offloading process, using the format conversion application for video files that converts any given format to .mp4 standard. The reason behind using this application is that, the performance bottleneck can be easily identified, where most of the process involves complex algorithms that are to be computed for every frame, and the pixel of information in the video. The hardware we use in the evaluation is listed in **Table 2**.

Table 2. Hardware components of Mobile Device and Resource Rich Server

Hardware	Client Device	RM Server	Wi-Fi Router
Processor	ARM	Intel i3 dual core	Broadcom
Speed	400 MHz	3.2GHz	54Mbps
Memory	256MB	2GB	Nil
Connectivity	Wi-Fi/2G	Wi-Fi/ ADSL	IEEE 802.11 b
Operating Sys.	Android 4.0	Windows 7	Dell WLAN A23 5.10

5.1. Results and Analysis

The following is the sample test case for 10 video files, which are converted to the .mp4 format within the mobile device. In **Table 3**, the original file of the Audio Video Interleave (.avi) format with its size in bytes and its video length in minutes is given for reference.

Table 3. Original movie file properties

S. No.	File Name	Size	Video Length(min)
1	Session.avi	18.76 MB	00:03:35
2	Two.avi	146.19 MB	00:06:45
3	Athu.avi	225.18 MB	00:18:07
4	Friend.avi	322.82 MB	00:26:01
5	Hai.avi	456.32 MB	00:21:23
6	David.avi	504.81 MB	00:40:38
7	Lost.avi	648.06 MB	00:52:10
8	Veera.avi	706.48 MB	00:56:52
9	Linkin.avi	881.78 MB	01:11:01
10	Jodi.avi	1.06 GB	01:21:48

The time taken for conversion to the .mp4 format, its battery consumed in percentage, and the size it occupies in the external storage space of the mobile after conversion are listed in table 4. For conversion purpose, the rate at which the number of frames per second is being converted is considered for evaluation. Also, we have taken three different test case scenarios, in terms of quality, that is, the resolution of pixel value with 480 x 360, which is a downsizing of the quality. The second test case has the original quality of 960 x 720 as it is, and the third has a higher quality pixel ratio of 1024 x 768, that is of HD quality. The preliminary analysis clearly shows that, as the size of the file increases, the time to execute or convert the original file format from .avi to .mp4 increases proportionally. This, in-turn, fuels the battery power to drain accordingly, and utilizes the maximum energy. The Lowest resolution of 480 x 360 is converted using mobile and its battery usage is noted in **Table 4**.

Table 4. Conversion within the mobile at 480 x 360 resolutions

File Name	Size	Frames/ Second	Time Taken	Battery (%)
Session.mp4	23.62 MB	17	00:04:26	2
Two.mp4	49.91 MB	10	00:21:46	5
Athu.mp4	124 MB	12	00:41:06	8
Friend.mp4	177 MB	11	00:35:25	7
Hai.mp4	148 MB	13	00:59:00	12
David.mp4	278 MB	12	01:24:36	19
Lost.mp4	357 MB	11	01:53:35	21
Veera.mp4	389 MB	12	01:58:20	23
Linkin.mp4	486 MB	11	02:41:12	40
Jodi.mp4	620 MB	12	03.04.34	45

Table 5, shows that the time taken for converting to the original resolution is more, in accordance with the size of the file.

Table 5. Conversion within the mobile at 960 x 720 resolutions

File Name	Size	Frames/ Second	Time Taken	Battery (%)
Session.mp4	24.18 MB	7	00:22:10	3
Two.mp4	52.95 MB	10	00:20:15	4
Athu.mp4	132 MB	14	00:34:40	7
Friend.mp4	178 MB	15	00:27:50	6
Hai.mp4	151 MB	8	01:10:20	13
David.mp4	278 MB	14	01:12:30	15
Lost.mp4	415 MB	13	01:44:03	20
Veera.mp4	403 MB	13	01:41:30	18
Linkin.mp4	0.94 GB	10	02:41:50	39
Jodi.mp4	1.2 GB	12	03.15.45	44

The results of **Table 6,** shows that a heavier file certainly drains the battery power.

Table 6. Conversion within the mobile at 1024 x 768 resolutions

File Name	Size	Frames/ Second	Time Taken	Battery (%)
Session.mp4	24.18 MB	7	00:12:42	3
Two.mp4	57.29 MB	5	00:40:24	7
Athu.mp4	143 MB	5	01:30:30	21
Friend.mp4	178 MB	5	01:18:60	19
Hai.mp4	154 MB	5	02:08:10	26
David.mp4	278 MB	5	03:23:03	45
Lost.mp4	448 MB	5	04:09:60	63
Veera.mp4	433 MB	6	03:56:30	48
Linkin.mp4	1.02 GB	5	05:54:55	65
Jodi.mp4	1.35 GB	5	06.55.45	78

Table 7 below depicts the conversion process being undertaken in the server at a resolution of 960 x 720.The result shows that the time taken for conversion is the same as the length of the video file, and the size is almost double that of the original file.

Table 7. Conversion in the Offloaded server at 960 x 720 resolutions

File Name	Size	Video Length	Time Taken
Session.mp4	44.79 MB	00:03:35	00:03:35
Two.mp4	238 MB	00:06:45	00:06:45
Athu.mp4	581 MB	00:18:07	00:18:07
Friend.mp4	330 MB	00:26:01	00:26:01
Hai.mp4	557 MB	00:21:23	00:21:23
David.mp4	0.99 GB	00:40:38	00:40:38
Lost.mp4	1.72 GB	00:52:10	00:52:10
Veera.mp4	1.76 GB	00:56:52	00:56:52
Linkin.mp4	3.12 GB	01:11:01	01:11:01
Jodi.mp4	2.24 GB	01:21:48	01:21:48

Even though the time for conversion is dependent on the running time of the video, the total time for offloading to the server, involves the transmission and receipt of the file. This process of to and fro transmission is directly proportional to the size of the file, which has been recorded in **Table 8**.

Table 8. Transfer time between the Mobile and the Server

File Name	Original File Transfer from the Mobile to the Server			Converted File Transfer from the Server to the Mobile		
	Size	Time Taken	Battery (%)	Size	Time Taken	Battery (%)
Session.mp4	18.76 MB	00:00:05	0.1	44.79 MB	00.00.11	0.5
Two.mp4	146.19 MB	00:00:19	0.4	238 MB	00.02.15	1
Athu.mp4	225.18 MB	00:02:10	1	581 MB	00.04.20	1.5
Friend.mp4	322.82 MB	00:03:00	1	330 MB	00.03.12	1
Hai.mp4	456.32 MB	00:03:50	1	557 MB	00.04.13	1.5
David.mp4	504.81 MB	00:04:13	1.5	0.99 GB	00.07.20	2
Lost.mp4	648.06 MB	00:05:01	2	1.72 GB	00.11.45	3
Veera.mp4	706.48 MB	00:05:15	2	1.76 GB	00.11.50	3
Linkin.mp4	881.78 MB	00:06:34	2	3.12 GB	00.20.46	5
Jodi.mp4	1.06 GB	00.07.33	2	2.24 GB	00.16.57	4

The total time taken for transferring a file for conversion and receiving it back, is approximately between 15 seconds to 26 minutes at the most. The battery consumption for relaying is much less for a mobile, in the range of 0.5 to 7 percent. Here, in our implementation scenario, we have not considered the power utilized by the server for its computation and transmission, because its consumption does not critically impact process of offloading. The final analysis in **Table 9** describes how efficiently the time and power are utilized when offloading is carried out, by just adding the time and battery power of the earlier analysis.

Table 9. Comparative analysis of Mobile and Server Computation for Time and Power

File Name	Computation Done Within the Mobile			Computation Done in the Server		
	Total Size Occupied in Mobile in MB	Total Time	Battery (%)	Total Data Transfer Size in MB	Total Time Taken	Battery (%)
Session.mp4	42.94	00:22:10	3	63.55	00.03.16	0.6
Two.mp4	199.14	00:20:15	4	384.19	00.09.19	1.4
Athu.mp4	357.18	00:34:40	7	806.18	00.24.37	2.5
Friend.mp4	500.82	00:27:50	6	652.82	00.32.13	2
Hai.mp4	607.32	01:10:20	13	1013.32	00.29.26	2.5
David.mp4	782.81	01:12:30	15	1494.81	00.52.01	3.5
Lost.mp4	1063.06	01:44:03	20	2368.06	01.08.56	5
Veera.mp4	1109.48	01:41:30	18	2466.48	01.13.57	5
Linkin.mp4	1821.78	02:41:50	39	4001.78	01.38.21	7
Jodi.mp4	2260	03.15.45	44	3300	01.46.18	6

The analysis results play an important role in proving the efficiency of offloading. The computation in the server proves to be more efficient when compared to the computation in the mobile device itself. The graph in **Fig. 5** refers to the performance of the proposed algorithm. The testing is done with various sizes of the video file, and the graph shows in its Y-axis the time duration for the conversion, and in the X-axis the size of video file.

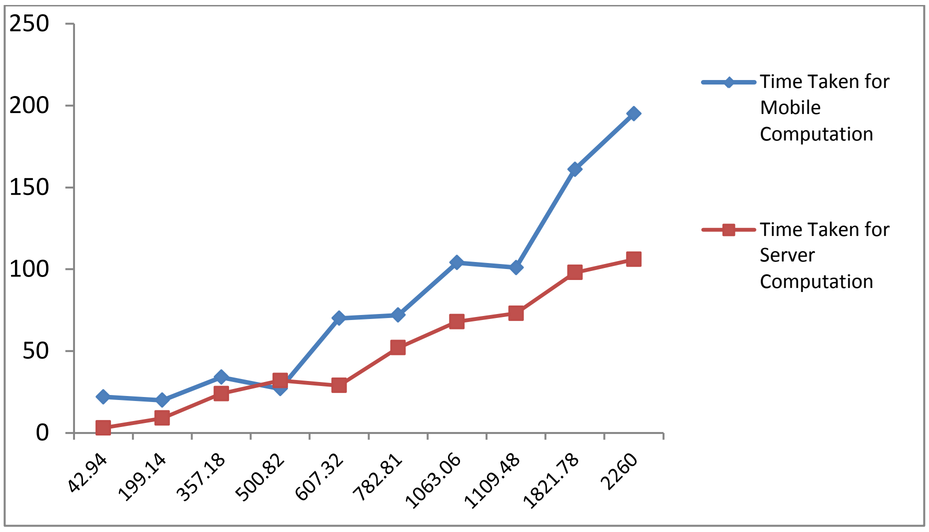

Fig. 5. Analysis of the Conversion time within the mobile and the server

Thus, the values show that the offloading to the server takes less time, which stresses the need for offloading to save time and computational process in mobile. The analysis in **Fig. 6** shows the graph for battery consumption, in terms of percentage in the X-axis, and the corresponding time duration for computation in the Y-axis.

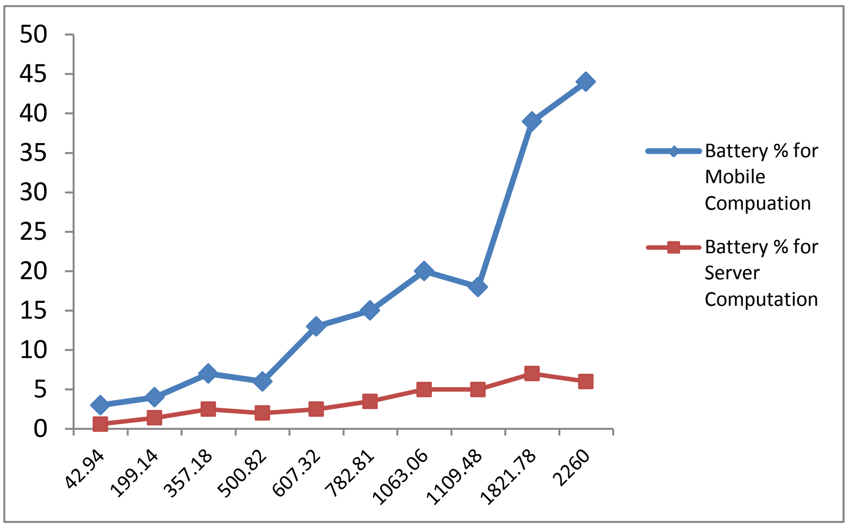

Fig. 6. Battery Performance between the mobile and Offloading Server Computation

The result shows that for the server offloading process, the battery consumed is very less, that is within 5 to 7 percent, whereas for computation in the mobile it takes more than 25 percentage of battery power. Thus, it proves that the offloading strategy is an efficient way to do computation, when the mobile is constrained in power and processing capabilities.

Table 10. Comparative Analysis of the Existing and Proposed Offloading Decision Schemes

Offloading Decision Schemes	**Environment**	**Application Type**	**Cloud Migration**	**Static/Dynamic Adaptation**	**N/W Load**	**Decision Attributes**
Proposed RM Server	Android	Video Conversion	Yes	Dynamic & Static	Very High	Client and Context
VM based Cloudlet [2]	Virtual Machine	KPresenter, GIMP	No	Static	Low	Bandwidth
Hyrax [7]	Hadoop	Multimedia Searching	No	Static	Very High	Network Limitations
MAUI [19]	.NET	Face Recognition	No	Dynamic	Very Low	RTT, Energy
Energy-Aware [20]	Java, RMI	Image Processing	No	Static	Very Low	Energy
Weblets [21]	C#	Augmented Reality	Yes	Static & Dynamic	High	Latency

The analysis of similar offloading decision schemes given in **Table 10**, clearly prove that our proposed computation architecture provide better service, in terms of high network load in a dynamic adaptation policy. The decision to offload is primarily based on the client and the context-aware capabilities in our system, whereas most of the other schemes are related only to the network and power. Support for cloud migration is another important criterion to weigh our system, which is essentially the future of mobile computing. Nowadays, many cloud servers offer their service at very minimal or no cost; hence, mobile cloud computing could well be the best choice for computation offloading.

5.2. Future Directions

Computation offloading is a future technique, with a simple solution to improve execution efficiency of a mobile based system. Some of the real world applications that could be made possible through offloading are, rendering a 3D animation, Natural Language Processing (NLP) and Image Processing. In the first case, let us assume having a simple 3D animator app in the mobile, which is used to create quick models and animations. In order to complete the process of animation, we need rendering, which is a complex time and memory consuming task when executed in any mobile device. It requires Graphics Processing Unit (GPU) based computer system, which is found in server configurations. Thus, a local execution in a nearby server through offloading via, Wi-Fi medium, will quickly serve the purpose of rendering the animation to a movie file and deliver it back to the creator. In the second example, for language processing, assume Optical Character Recognition (OCR) software processing multiple images sent from various mobile devices connected through wireless medium in a typical collaborative learning environment. Any Natural Language Processing requires complex algorithms to be computed to execute even a simple task, which is not a feasible option with mobile computing. Since, the source of data for NLP could be easily captured using a mobile phone camera or any digital camera enabled with wireless communication. These devices can communicate for getting serviced with the offloading server for computation. Thus the incompetent processing power of the mobile could be offloaded for efficient utilization. In the third case, when an image processing is done, there are multiple processes like segmentation and feature extraction. It requires software like MatLab, to do the processes for the input and extract the desired output. The future scope of image processing like image mining and video search technique may be built as a thin client mobile app with just a user interface, instead of developing it as native application. These interfaces will always require a nearby server to do all the computation, thereby utilizing the compute power, memory, and system load and as well, satisfy the most important issue of battery life. Finally, the case of offloading need not be always with a local RM server. In future, cloud connected mobile application will be the default choice in every operating system running the mobile device.

6. Conclusion

The thrust towards the mobile cloud is greater than ever, as the need for volume-based computation by the mobile community is susceptible to dearth of resources. This issue is addressed by offloading computation for seamless interaction and integration of multiple resources, supporting the needs of mobile devices. Our approach to provide an analysis and migration services, that would indeed require offloading in a context and client aware environment, have proved to be successful. The time and power analysis that are estimated, by comparing the video file format conversion application in the mobile and cloudlet, have helped us to understand the choice of having a nearby resource-rich server, offering service to location based clients. To further extend the connectivity, the cloudlet holds the attributes of multiple clients, to serve them better by offering solutions like computation and storage from the public cloud. Henceforth, mobile cloud computing will become a good choice, for communication to shift the paradigm to the next higher level, which integrates pervasive nature to cloud access.

References

[1] Karthik Kumar, Jibang Liu, Yung Hsiang Lu and Bharat Bhargav, "A survey of computation offloading for mobile systems," *Mobile Network Application*, *Springer*, vol. 18, no. 1, pp. 129-140, February, 2013. Article (CrossRef Link).

[2] Mahadev Satyanarayanan, Bahl P, Caceres R, Nigel Davies, "The Case for VM-based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14-23, December, 2010. Article (CrossRef Link).

[3] Bo Han, Madhav V. Marathe, Jianhua Shao, Aravind Srinivasan, "Mobile Data Offloading through Opportunistic Communications and Social Participation," *IEEE Transaction on Mobile Computing*, vol. 11, no. 5, pp. 821-834, May, 2012. Article (CrossRef Link).

[4] Pelin Angin and Bharat K. Bhargava, "Real-time Mobile Cloud Computing for Context-Aware Blind Navigation," *International Journal of Next-Generation Computing*, vol. 2, no. 2, 2011. Article (CrossRef Link).

[5] Van Vinh Nguyen and Jong Weon Lee, "A Hybrid Positioning System for Indoor Navigation on Mobile Phones using Panoramic Images," *Transaction on Internet and Information Systems*, vol. 6, no. 3, pp. 835-850, March, 2012. Article (CrossRef Link).

[6] Dong Huang, Ping Wang and Dusit Niyato, "A Dynamic Offloading Algorithm for Mobile Computing," *IEEE transactions on wireless communication*, vol. 11, no. 6, pp. 1991-1995, June, 2012. Article (CrossRef Link).

[7] E.E.Marinelli, "Hyrax: cloud computing on mobile devices using Map Reduce, "*DTIC Document*, *Tech.Rep.*," September, 2009. Article (CrossRef Link).

[8] Sun-Rae Park et.al., "Intelligent u-Learning and Research Environment for Computational Science on Mobile Device," *Transaction on Internet and Information Systems*, vol. 8, no. 2, pp. 707-720, February 2014. Article (CrossRef Link)

[9] A. Klein, C. Mannweiler, J. Schneider, and H. D. Schotten, "Access schemes for mobile cloud computing," in *Proc. of IEEE Eleventh International Conference on Mobile Data Management (MDM)*, pp. 387–392, 2010. Article (CrossRef Link).

[10] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587-1611, December, 2011. Article (CrossRef Link).

[11] L. Guan, X. Ke, M. Song, and J. Song, "A survey of research on mobile cloud computing," in *Proc. of IEEE/ACIS 10th International Conference on Computer and Information Science (ICIS)*, pp. 387– 392, 2011. Article (CrossRef Link).

[12] D. Kovachev, Y. Cao, and R. Klamma, "Mobile Cloud Computing: A Comparison of Application Models," *arXiv Cornell University Library*, July 2011. Article (CrossRef Link).

[13] Fuhong Lin, Xianewi Zhou, Changjia Chen, "Novel Pre-pushing & Downloading Model in Mobile Peer-assisted Streaming Network," *Transaction on Internet and Information Systems*, vol. 7, no. 12, pp. 3135-3148, December 2013. Article (CrossRef Link).

[14] Ji Yao, Jiguo Li and Yichen Zhang, "Certificate-Based Encryption Scheme without Pairing," *Transaction on Internet and Information Systems*, vol. 7, no. 6, pp. 1480-1491, June 2013. Article (CrossRef Link).

[15] Md. Sabbir Hasan, Eui-Nam Huh, "Heuristic based Energy-aware Resource Allocation by Dynamic Consolidation of Virtual Machines in Cloud Data Center," *Transaction on Internet and Information Systems*, vol. 7, no. 8, pp. 1825-1842, August 2013. Article (CrossRef Link).

[16] Young Bae Yoon, Junseok Oh and Bong Gyou Lee, "The Establishment of Security Strategies for Introducing Cloud Computing," *Transaction on Internet and Information Systems*, vol. 7, no. 4, pp. 860-867,April 2013. Article (CrossRef Link).

[17] Chenlei Cao, Ru Zhang, Mengyi Zhang and Yixian Yang, "IBC-Based Entity Authentication Protocols for Federated Cloud Systems," *Transaction on Internet and Information Systems*, vol. 7, no. 5, pp. 1291-1312, May 2013. Article (CrossRef Link).

[18] Leyou Zhang, Yupu Hu, "New Constructions of Hierarchical Attribute-Based Encryption for Fine-Grained Access Control in Cloud Computing," *Transaction on Internet and Information Systems*, vol. 7, no. 5, pp. 1343-1356, May 2013. Article (CrossRef Link).

[19] Cuervo E, Balasubramanian A, Cho D, Wolman A, SaroiuS, Chandra R, Bahl P, " MAUI: making smartphones last longer with code offload," in *Proc. of International conference on mobile systems, applications, and services*, pp 49–62, 2010. Article (CrossRef Link).

[20] Chen G, Kang B-T, Kandemir M, Vijaykrishnan N, Irwin MJ, Chandramouli R, "Studying energy tradeoffs in offloading computation/compilation in java-enabled mobile devices," *IEEE Transaction on Parallel Distributed System*, vol. 15, no.9, pp. 795–809, September 2004. Article (CrossRef Link).

[21] X. Zhang, S. Jeong, A. Kunjithapatham, and Simon Gibbs, "Towards an Elastic Application Model for Augmenting Computing Capabilities of Mobile Platforms," in *Proc. of Third International ICST Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, LNICST 48, Springer, pp. 161-174, July 2010. Article (CrossRef Link).

[22] Hyukho Kim, Woongsup Kim and Yangwoo Kim, "A Pattern-Based Prediction Model for Dynamic Resource Provisioning in Cloud Environment," *Transaction on Internet and Information Systems*, vol. 5, no. 10, pp. 1712-1732, October 2011. Article (CrossRef Link).

[23] Romeo Mark A. Mateo and Jaewan Lee, "Dynamic Service Assignment based on Proportional Ordering for the Adaptive Resource Management of Cloud Systems," *Transaction on Internet and Information Systems*, vol. 5, no. 12, pp. 2294-2314, December 2013. Article (CrossRef Link).

Uma Nandhini has obtained her B.E degree in the field of Computer Science and Engineering from Bharathidasan University, India in 2002, and also obtained her Master's Degree in Computer Science and Engineering from Anna University, India in 2007. She is currently working towards her Ph.D. in the field of Information Technology from B.S.Abdur Rahman University, India. Her major research area includes, Cloud Computing, Mobile Communication and Energy Aware Computing.

Latha Tamilselvan has obtained her B.E degree in Electronics & Communication Engineering from Bharathidasan University, India in 1990, M.E in Communication Systems from Regional Engineering College, Bharathidasan University, India in 1995 and Ph.D. in Computer Science and Engineering from Anna University, India in 2008. Her research interest includes Cloud Computing, Mobile Ad hoc Network and Network Sec