J Grid Computing
DOI 10.1007/s10723-014-9323-6

Energy Efficient Computational Offloading Framework for Mobile Cloud Computing

Muhammad Shiraz · Abdullah Gani · Azra Shamim · Suleman Khan · Raja Wasim Ahmad

Received: 8 May 2013 / Accepted: 9 December 2014

Abstract The latest developments in mobile computing technology have changed user preferences for computing. However, in spite of all the advancements in the recent years, Smart Mobile Devices (SMDs) are still low potential computing devices which are limited in memory capacity, CPU speed and battery power lifetime. Therefore, Mobile Cloud Computing (MCC) employs computational offloading for enabling computationally intensive mobile applications on SMDs. However, state-of-the-art computational offloading frameworks lack of considering the additional overhead of components migration at runtime. Therefore resources intensive and energy consuming distributed application execution platform is established. This paper proposes a novel distributed Energy Efficient Computational Offloading Framework (EECOF) for the processing of intensive mobile applications in MCC. The framework focuses on leveraging application processing services of cloud datacenters with minimal instances of computationally intensive component migration at runtime. As a result, the size of data transmission and energy consumption cost is reduced in computational offloading for MCC. We evaluate the proposed framework by benchmarking prototype application in the real MCC environment. Analysis of the results show that by employing EECOF the size of data transmission over the wireless network medium is reduced by 84 % and energy consumption cost is reduced by 69.9 % in offloading different components of the prototype application. Hence, EECOF provides an energy efficient application layer solution for computational offloading in MCC.

Keywords Mobile cloud computing · Distributed systems · Computational offloading · Lightweight · Energy efficient

M. Shiraz (✉) · A. Gani · A. Shamim · S. Khan · R. W. Ahmad
Centre for Mobile Cloud Computing Research (C4MCCR), Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia
e-mail: muh_shiraz@um.edu.my

A. Gani
e-mail: abdullah@um.edu.my

A. Shamim
e-mail: azra.majeed864@yahoo.com

S. Khan
e-mail: suleman.khan.dr@ieee.org

R. W. Ahmad
e-mail: wasimraja@siswa.um.edu.my

1 Introduction

The latest developments in mobile computing technology have changed user preferences for computing. The report of Juniper Research states that consumer and enterprise market for cloud based mobile applications is expected to raise $9.5 billion by 2014 [1], which is

Published online: 13 January 2015

an evidence of the increasing use of MCC. Recently, a number of computing and communication devices are replaced by smartphones towards all-in-one ubiquitous computing devices such as PDAs, digital cameras, Internet browsing devices, and Global Positioning Systems (GPS) [2]. Human dependency on the contemporary smartphones is increasing rapidly in various domains such as enterprise, e-learning and entertainment, gamming, management information systems, and healthcare [3]. Mobile devices are predicated as the dominant future computing devices with high user expectations for accessing computational intensive applications analogous to the powerful stationary computing machines. However, in spite of all the advancements in the recent years, SMDs are still low potential computing devices and mobile applications on the latest generation of smartphones and tablets are constraint by battery power, CPU potentials and memory capacity of the SMDs [4]. The latest developments in cloud computing facilitates to increase the computing capabilities of resources constrained client devices by offering leased infrastructure and services of clouds [5–11]. Computational clouds employ different IT business models for the provisioning of computing services; such as on-demand, pay-as-you-go, and utility computing [6]. For example, Amazon Web Services (AWS) are utilized to store personal data through its Simple Storage Service (S3) [12], and Elastic Cloud Compute (EC2) is employed for application processing services. MCC enables computationally intensive and ubiquitous mobile applications by leveraging the services of computational clouds.

MCC utilizes the application processing services of computational clouds for the processing of computationally intensive mobile applications. Recently, a number of Computational Offloading Frameworks (COFs) are proposed for the processing of intensive mobile applications in MCC [13–19]. For instance Apple iCloud [20] and Amazon Silk [21] browser are two latest mobile applications which leverage the services of computational cloud for application processing. However, state-of-the-art computational offloading frameworks employ adaptive algorithm in which elastic mobile applications dynamically distribute intensive partitions to the cloud server nodes [23]. Such frameworks focus on offloading intensive mobile applications at different granularity levels and establishing distributed application processing platform at runtime. Runtime computational offloading involves cost of migration of the intensive components of the mobile application [13–19, 23], which involves computing resources utilization in transferring the application binary file and corresponding data file of the running instances of mobile application. Similarly, a number of application offloading frameworks implement dynamic application profiling and partitioning technique for application offloading [13–19], which increases the turnaround time of the application and energy consumption on mobile device. Current COFs focus on what components of the application to offload, how to offload and where to offload the intensive components of the application. Traditional computational offloading frameworks lack of considering the intensity of runtime component offloading and focus on leveraging the IaaS model for computational offloading which is resources intensive and time consuming [13, 17–19]. Therefore, such frameworks employ energy consuming procedure for computational offloading in MCC.

In this paper, we propose an Energy Efficient Computational Offloading Framework (EECOF) for computational offloading in MCC. The framework focuses on leveraging application processing services of cloud datacenters with minimal instances of application migration at runtime. As a result, energy consumption cost is reduced in remote processing of computationally intensive components of the mobile application. EECOF incorporates the Software-as-a-Service (SaaS) model and the Infrastructure-as-a-Service (IaaS) model of computational clouds for leveraging the application processing services of computational clouds. The proposed framework is evaluated by benchmarking prototype mobile application in the real MCC environment. Experimentation is performed in three different scenarios: (a) execution of mobile application on local mobile device, (b) implementing traditional computational offloading technique [15, 42] for the execution of application and (c) employing EECOF for computational offloading in MCC. Energy efficiency is measured in terms of reduction of energy consumption cost for computational offloading in MCC. The energy efficient nature of EECOF is validated by comparing results of different experimental scenarios. It is examined that by employing EECOF the overhead of runtime application migration is minimized, which results in the reduction of energy consumption cost for

computational offloading. Analysis indicates that the size of data transmission over the wireless network medium is reduced up to 84 % and energy consumption cost is reduced up to 69.9 % while offloading different components of the prototype application. Hence, the employment of EECOF provides an energy efficient application layer solution for computational offloading in MCC.

The paper is classified into the following sections. Section 2 discusses state-of-the-art computational offloading frameworks for MCC. Section 3 presents the architecture of proposed framework and discusses the operating procedure of EECOF and presents the distinctive features of the proposed framework. Section 4 discusses methodology used for the evaluation of proposed framework. Section 5 presents results and discusses experimental findings. Finally, Section 6 draws concluding remarks and future directions.

2 Related Work

Satyanarayanan [24, 25] introduced the term cyber foraging to augment the computing potentials of wireless mobile devices by utilizing available stationary computers in the local environment. Earlier, application offloading was employed for Pervasive computing [26], Grid computing [27] and Cluster computing [28]. Recently, a number of cloud server based application offloading frameworks are proposed for outsourcing computational intensive mobile applications partially or entirely to cloud datacenters [13–19]. Elastic applications are partitioned at runtime for the establishment of distributed processing platform [4]. Current frameworks implement computational offloading at different granularity levels such as object, class, component, bundle, thread, method and task. In [17, 19–21, 42] module level offloading is employed for the migration of entire module of the application. In [13] method level granularity is employed for the migration of pre-annotated intensive methods of the application. In [29] object level granularity is used for dynamic computational offloading. In [30, 31] thread level granularity is used for separating the intensive logic of the application at runtime. In [32, 33] class level granularity is employed for runtime computational offloading. In [34, 35] component level partitioning is employed which indicates that a group of classes are offloaded to the remote server at runtime. COFs implement application partitioning statically or dynamically. The static application partitioning [36] involves one time application partitioning mechanism for the distribution of workload between SMD and cloud server node. The intensive components of the application are partitioned and transferred to the remote server node. For example, the primary functionality offloading [35] mechanism involves partitioning and offloading of the intensive components at runtime.

In [17] a middleware framework is proposed for the dynamic distribution of application processing load between SMD and cloud server node. The framework deploys the application partitioning in optimal mode and dynamically determines the execution location for modules of the mobile application. In order to reduce search space, the framework implements a preprocessing mechanism on the consumption graph. Preprocessing separates local and remote bundles of the application. It searches for the application modules which can result in high cost in offloading and for that reason are not feasible for offloading. The framework distributes workload as per the statistics of resources available on SMD. The distributed platform in AIDE [16] is composed of computing devices in the locality of mobile devices. SMD access a preconfigured surrogate server which maintains information of the volunteer server nodes. The runtime profiling mechanism of the framework is activated dynamically which implements class level granularity for partitioning of mobile application.

Mobile Assistance Using Infrastructure (MAUI) [13] focuses on energy saving for SMD. Application developers identify the local and remote components of the application at design time. The MAUI profiler determines the feasibility of remotely annotated method for offload processing. Each time a method is called, the profiler component assesses it for energy saving which utilizes additional computing resources (CPU, battery) on SMD. MAUI solver operates on the input provided by application profiler. It determines the destination of execution for the method annotated as remote. MAUI implements application level partitioning for outsourcing computational load of SMD. However, the mechanism of runtime application profiling and solving at runtime involves additional computing resources utilization for application partitioning. Development of the applications on the basis of MAUI requires additional developmental efforts

for annotating the execution pattern of each individual method the application. MAUI involves the overhead of dynamic application profiling, solving, partitioning, migration, and reintegration on SMD. In CloneCloud [37] the partitioning and reintegration of the application occur at application level. Partitioning phase of the framework includes static analysis, dynamic application profiling, and optimization solution. A preprocess migratory thread is implemented on mobile devices to assist in the partitioning and reintegration of the thread states. Elastic application model [14] provides a middleware framework for mobile applications. Application is dynamically partitioned into weblets which are migrated dynamically to cloud server node. The framework implements different elastic patterns for the replication of weblets on the remote cloud. It considers different parameters for offloading of the weblets; such as status of the mobile device, cloud, application performance measures and user preferences which comprise power saving mode, high speed mode, low cost mode and offload mode. The framework implements a resources intensive mechanism for runtime application partitioning and the migration of weblets between SMD and remote cloud nodes. It includes additional resources utilization on SMD in the process of application profiling, dynamic runtime partitioning, weblets migration and reintegration, and continuous synchronization with cloud server node for the entire duration of application processing.

Current COFs accomplish distributed application processing platform for the processing of intensive mobile applications in diverse modes. Several approaches exploit VM migration [13, 23]; others focus on part(s) of the application to be offloaded [13, 17, 21]. A number of approaches implement dynamic application partitioning [13, 14], whereas other focus on static partitioning [36, 37]. Diverse objective functions are considered [4]; saving processing power, efficient bandwidth utilization, saving energy consumption [38], user preferences and execution cost [14]. Recently, a number of mobile cloud applications are witnessed which employ cloud computing to mitigate resources constraints of SMDs. Apple's iCloud [20] provides on demand access automatically to applications such as music, photos, apps, calendars, documents. Amazon EC2 and Microsoft Azure host the application store of Apple's iCloud. Similarly, Amazon released Silk application [21] which is a cloud-accelerated web browser Silk is a "split browser" which resides on both Kindle Fire and EC2. For each web page request, Silk dynamically determines distribution of computational load between the local SMD and remote Amazon EC2. Silk considers the objective functions of network conditions, page complexity and the location of any cached content [21].

In traditional computational offloading frameworks cloud based application processing is composed of three phases which include offloading initialization, computational offloading and remote application execution. (a) In the initialization phase, the availability of services on the cloud server node are discovered, network and mobile context information reports are collected from the corresponding sensor modules. Furthermore, user preference and application characteristics such as security level and QoS demands are also gathered. The information collected in this phase is used for the offloading mechanisms. (b) In the computational offloading process decisions are taken and parameter configurations are performed. This phase includes decision of application partitioning and offloading of an application, user authentication and authorization, VM instance creation on mobile and cloud server if supported by an execution framework, migration of VM clone, QoS parameter negotiation, SLA exchange, resource reservation and admission control. (c) Once the delegated application is configured the running state of the application is resumed on the remote virtual device instance and application is executed on remote server node. The initiation and preparation overhead is required to be reduced so that application faces minimum disruption during the execution due to application of?oading.

Traditional COFs employ computational offloading at runtime which requires virtual device instances on cloud server node by employing IaaS services of computational clouds [13, 17–19]. The binary files of the application and the data states of the running application are offloaded at runtime which increase the energy consumption cost of computational offloading for MCC. The Energy consumption Cost (E_c) for each component offloaded at runtime includes energy consumed in runtime component migration (E_m), energy consumed in saving the data states of running instance of the mobile application (E_s), energy consumed in uploading the data file to remote server node (E_u) and energy consumed in returning the resultant data files

to local mobile device (E_d). Hence, the total energy consumption cost for each component offloaded at runtime is represented by equation (1).

$$E_c = E_m + E_s + E_u + E_d \tag{1}$$

Let E represents the set of energy consumption cost of all instances $Ec_{a=1,2,\ldots,n}$ of the runtime component offloading and E is the cardinality of set E, which shows energy consumption cost of the total instances of runtime component offloading. The total energy consumption of the runtime application offloading is represented by μ, which is shown as follows.

$$\mu = \sum_{a=1}^{n} Ec_a \qquad \Rightarrow \forall Ec_a \in E \wedge |E| \geq 1 \tag{2}$$

3 Proposed Energy Efficient Computational Offloading Framework (EECOF)

We propose an energy efficient computational offloading framework for the processing of intensive mobile applications in MCC. The framework focuses on leveraging application processing services of cloud datacenters with minimal instances of application code migration at runtime. The instances of runtime component migration are minimized by deploying computational task offloading as the primary offloading procedure rather than intensive component migration. As a result, energy consumption cost is reduced in remote processing of computationally intensive components of the mobile application. Traditional computational offloading frameworks [13, 17–19, 23] employ IaaS model for the deployment of virtual mobile device instances on the virtual machine of the cloud server node, which is resources intensive and energy starving mechanism. EECOF incorporates SaaS model with IaaS model of computational clouds for reducing the instances of runtime component migration.

The deployment of SaaS model eliminates energy consumption cost of runtime component migration (E_m), energy consumption cost of saving states of running mobile application (E_s) and energy consumption cost of uploading the data file to remote server node (E_u). Therefore, by employing EECOF the energy consumption cost is reduced in computational offloading. The advantages of the employing SaaS services for the configuration of intensive operations of the mobile application are mentioned as follows: (a) The overhead of runtime computation offloading involves the cost of application partitioning and cost of migration of binary code of the application and data states of the running instances of mobile application. Therefore, the configuration of resources intensive components of the mobile application on the cloud server nodes results in minimal instances of intensive component migration at runtime which reduces the overhead of computationally intensive components of the mobile application. (b) In traditional computational offloading frameworks [15, 42], the delegated computationally intensive components of the mobile application to the cloud server node are configured on the instances of virtual mobile device, which involves additional overhead of virtual device deployment and management on the cloud server node [35]. EECOF however, reduces this overhead by configuring remote services on the powerful virtual machines in the cloud datacenters rather than virtual devices instances of mobile devices, which decreases the turnaround time and energy consumption cost of the application. As a result, size of data transmission, computational resources utilization (RAM and CPU), energy consumption cost and turnaround time of the application is reduced.

Figure 1 shows the architecture of the proposed framework. Middleware on both mobile device and server is deployed for enabling the execution of mobile application in the primary operating procedure. Whereas, the components employed in the secondary operating procedure include orchestrator, preferences manager, upload manager and download manager. Synchronizer component is used for ensuring consistency in the distributed processing of application.

Orchestrator The orchestrator component of EECOF monitors switching between the operation modes (online/offline) of the mobile application and operation procedures of the online mode of the application. In the Primary Operating Procedure (POP) the orchestrator enables mobile application on the local device to access the preconfigured services, whereas in the Secondary Operating Procedure (SOP) preferences manager component is activated to save the data states of the running mobile application. Application orchestrator is responsible for the configuration

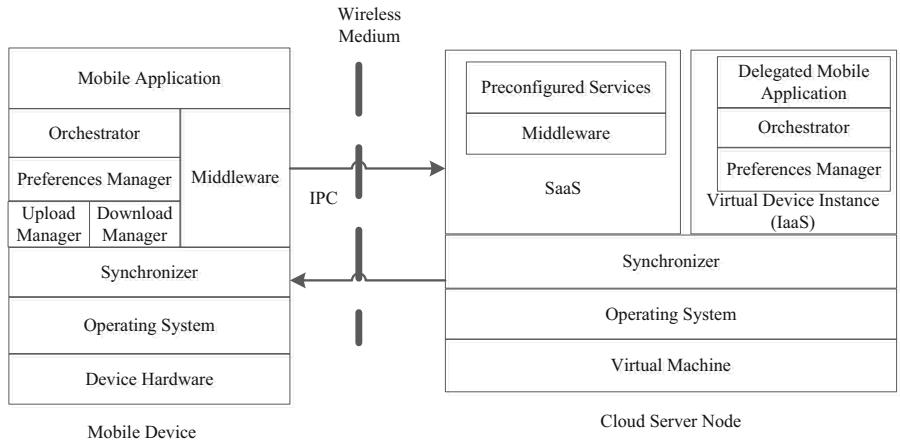

Fig. 1 Architecture of the proposed EECOF

of the mobile application on SMD and remote server node. On the cloud server node, application orchestrator configures the delegated service application on the remote server node. Application orchestrator component on the remote server node resumes the running state of the delegated mobile application by accessing the preferences files from the persistent storage. The application orchestrator component on the mobile device arbitrates with the remote server node for offloading the selected running component of the mobile application. The orchestrator component is assisted by the preferences manager component.

Preferences Manager Mobile applications are associated with a separated preferences manger which provides access to the preferences file. Preference manager reads and writes the data states from persistent storage during the activation and deactivation of mobile application. The preferences manager component copies the preferences file to the external storage device which is directly accessible for the upload manger and download manager component. The preferences manager component of the server node is responsible for providing access to the data files which are downloaded with the delegated service application. Similarly, whenever the service application completes execution on the remote server node, the preferences manger saves the final results to the preferences file. The upload manager component is responsible for uploading the preferences files of the application to remote server node in the SOP. The preferences files are stored on the persisted storage by the preferences manager which is accessible for upload manager. Whenever, the offloaded mobile application is installed on the remote virtual device instance, the synchronization manger component of the server connects to the upload manger component of the application running on mobile device and makes request for preferences file.

Upload Manager Upload manager sends the requested preferences file to the synchronizer of the server node. The download manager component on mobile device is responsible for downloading the preferences files of the application from the remote server node in the SOP. Whenever, the offloaded component of the application completes execution, download manager component of the EECOF client is connected to return the resultant preferences file.

Download Manager The synchronization manger component of the EECOF framework utilizes the services of download manager and upload manager for the synchronization of distributed application processing in the SOP EECOF framework. In the SOP, mobile application is enabled to offload the intensive components of the application to cloud datacenters which are executed on temporarily allocated server node. Once the execution of delegated component of the application is completed, download manager receives the resultant data file and saves it to the persistent storage of the local mobile device.

Synchronizer The synchronizer component of the framework is responsible for the synchronization of transmission between SMD and remote server node. In the POP of distributed processing, the synchronizer component is responsible for ensuring the

consistency of transmission between mobile application on SMD and the server application running on the cloud server node. In the SOP, whenever the states of the application are saved on the persisted medium, the synchronizer component is activated to offload the service application to remote server node. The orchestrator component searches for the configuration file of the identified intensive component on mobile device. Whenever, the configuration of the service application is validated, the synchronizer component is activated to outsource the configuration files to remote server node. Synchronizer component of the remote server node is activated to receive the delegated service application. Whenever, the configuration file of the delegated service application is received successfully on the remote server node, the orchestrator component of the server node is activated to configure the delegated service application and resume the running states from the preferences file. The synchronizer component is also responsible for the uploading and downloading of preferences files between mobile device and remote server node. In such scenario, the role of synchronizer is to coordinate between local mobile application and the offloaded components of the application. The primary responsibility of synchronizer is to ensure the consistency of transmissions between application executing on mobile device and server application.

Reliance on the preconfigured services of the cloud server nodes leads to the problem of dependency on the centralized services and reduces offline usability. Similarly, it leads to the employment of thin client applications like traditional web and email applications, wherein the processing logic of the application is hosted on the remote server nodes and client applications provide user interface. Therefore, the EECOF employs replication of the mobile application on mobile device and cloud server node and proposes two distinct operating modes (offline mode and online mode) for the execution of mobile application. Similarly, it proposes two different operating procedures for accessing the application processing services of computational clouds. EECOF enables mobile applications to operate in two distinct modes; online and offline. Offline mode indicates the availability of sufficient computing resources on the mobile device wherein the components of mobile application are scheduled for execution on local mobile device. The online mode indicates the critical condition of insufficient resources on local mobile device, wherein the computational load of the application is offloaded to cloud server node. The online mode is implemented by deploying two distinct operation procedures including POP and SOP. Mobile application is designed with the objectives to offload computational task instead of offloading application in the primary operating procedure and employ runtime application migration in the secondary operating procedure. In the offline mode all the components of mobile application are executed locally on SMD. Whereas in the online mode, mobile application is enabled to access the preconfigured services of EECOF server and offload the intensive components of the application on demand basis. The dual operation mode enables mobile application to operate with full functionalities in the situations of inaccessibility of remote services. Offloading active service to cloud server node involves complicated mechanism. However, mobile users remain unaware of the complications of the remote execution. EECOF employs middleware service for providing the notion as entire components of the mobile application are executed locally on SMD.

The preconfigured services of the cloud server are composed of the components of the mobile application which are computational intensive and do not require user interaction. Application on mobile device is an ordinary application with all the components available on local mobile device. However, two types of additional attributes are included in the ordinary mobile application. The profiler mechanism evaluates availability of resources (RAM, CPU, battery power) and accessibility of remote services for switching between online and offline mode. Mobile application is capable to operate in the two distinct operating procedures of the online mode. In the POP of online mode, the preconfigured services on the cloud server node (SaaS model) are accessed, whereas in the SOP the states of the running instance of the application are saved and the intensive components are offloaded at runtime. The runtime computational offloading mechanism is implemented by employing the IaaS model wherein platform specific virtual device instance is created on the cloud server node and the delegated mobile application is reconfigured. The primary operating procedure is significant for future computationally intensive mobile applications which need to be designed and developed on the basis of distributed feature of EECOF,

whereas the secondary operating procedure is imperative for the existing mobile applications which lack of deploying the distributed architecture. Hence, EECOF addresses the challenges of future computationally intensive mobile applications for MCC and provides support for offloading traditional mobile application at coarse granularity level. Figure 2 shows sequence of interaction between application running on mobile device and application running on the cloud server node in the POP. Mobile application is enabled to access the preconfigured services of cloud server node which involves direct communication pattern using middleware services for interaction between client application running on mobile device and server application running on remote server node. Application running on mobile device invokes services on the cloud server node and passes the required input data. Processing logic of the application is executed on the cloud server node and results are returned to the application running on the mobile device. Middleware hides complications of the communication between the local mobile application and cloud server application. The communication between local mobile application and remote server application is enabled by employing Inter Process Communication (IPC) mechanism such as RPC or RMI. The heterogeneous operating system platforms of mobile devices implement different middleware services [39]. For instance, we employed kSOAP2 API for accessing the preconfigured services on the clou d server node. kSOAP2 is a lightweight and efficient SOAP client library for the Android platform [43]. Computational task is offloaded to the cloud server node and upon the successful execution of the task on the cloud server node, resultant data is returned to mobile device. The overhead of sending the binary files of the application is eliminated in the POP. Therefore, the size of data transmission, energy consumption cost and turnaround time of the offloaded component is reduced accordingly.

Figure 3 shows sequence diagram for the SOP of online mode of EECOF. In the scenario of unavailability of appropriate remote server, SOP of the mobile application is activated, wherein the intensive components of the application are offloaded at runtime at the service level granularity. SOP of EECOF is employed for handling the dynamic processing load on the local mobile device. The profiler mechanism is activated to identify the larger intensive components of the application which needs to be offloaded to the remote server node. The orchestrator component of EECOF monitors transition between POP and SOP. The preferences manager component is activated to

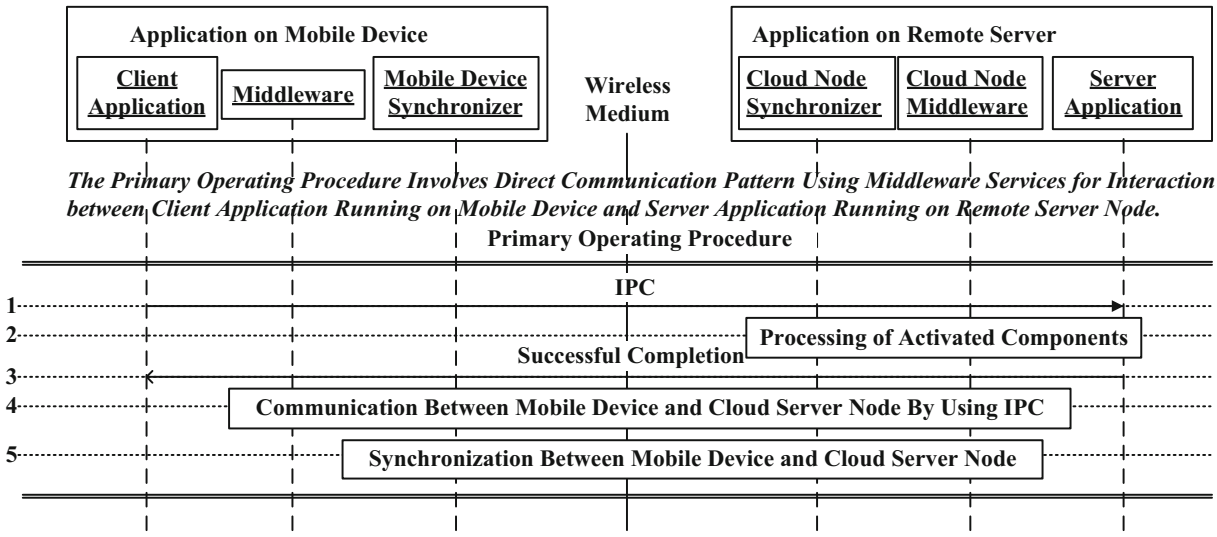

Fig. 2 Sequence diagram for the primary operating procedure of EECOF

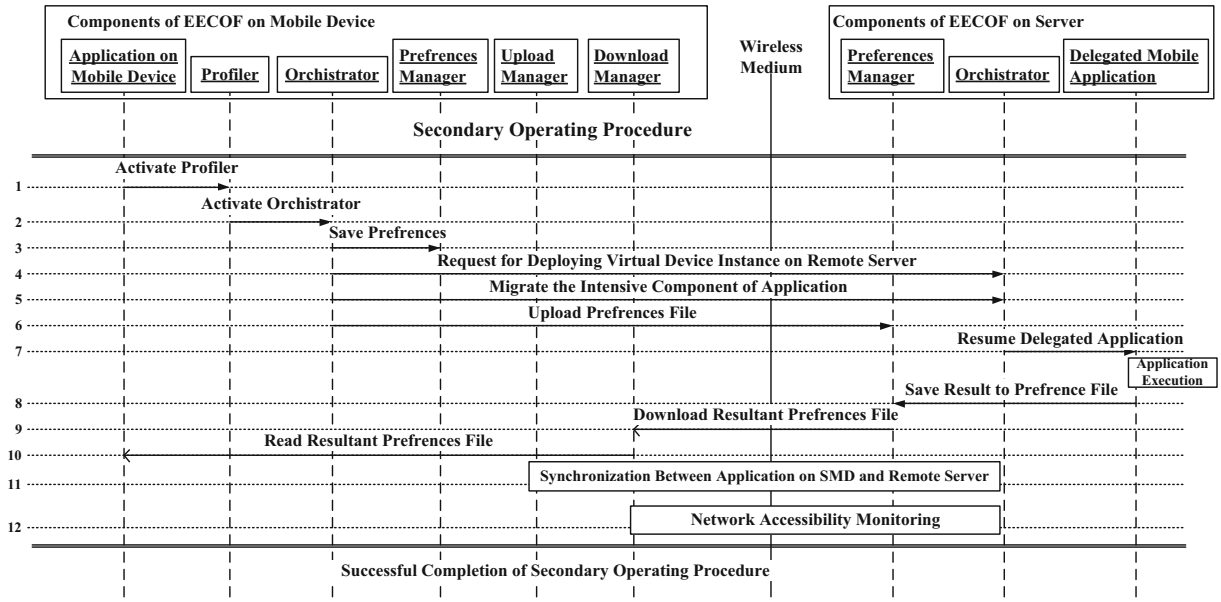

Fig. 3 Sequence diagram for the secondary operating procedure of EECOF

save the running states of the application. The state information of the intensive component of the application is saved in preferences file. In the mean while, the orchestrator component arbitrates with the cloud datacenter for the configuration of remote server node and virtual device instance. Following that, the identified intensive component of the application is offloaded to the remotely configured cloud server node. Virtual device instances on the cloud server node are inaccessible directly from external network environment; therefore, a proxy agent is configured on the cloud server node to forward the binary code the component of the application to virtual device instance. The orchestrator component on the cloud server node configures the delegated component of the mobile application. Following that, the preferences file of the application is uploaded which is copied to the data folder of the application by preferences manager component of the cloud server node. Preferences manager on the cloud server provides access to the uploaded preferences file of the application, from where the running state of the delegated application is resumed. The synchronizer component on the cloud server node is responsible for synchronizing the execution of application on the remote server node and application on mobile device. Final results are saved in the preferences file upon the successful execution of the application on the remote serve node. The resultant preferences file is downloaded to the mobile device by deploying the preferences downloader component of the ECCOF.

Figure 4 shows generic flowchart for the interaction of the components of EECOF framework in leveraging application processing services of cloud server node. Mobile application executes on SMD in two different modes, i.e. online and offline, whereas the services of remote server are utilized in two different operating procedures of online mode. In the online mode the services of cloud server nodes are leveraged for the processing of intensive components of the mobile application. In the POP, mobile application accesses the preconfigured services of EECOF server. However, in the scenario of unavailability of the preconfigured server, the intensive components of the mobile application are offloaded to the remote server node at runtime.

EECOF employs the replication of intensive components of the application on local mobile device and cloud server node. Application replication involves

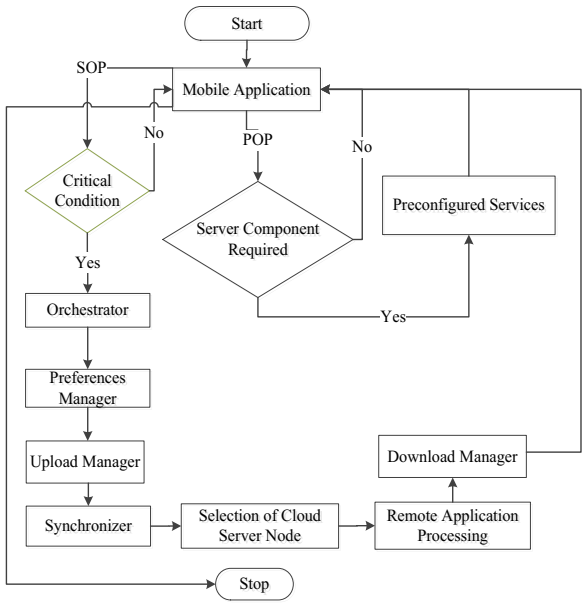

Fig. 4 Illustration of the interaction of the components of EECOF framework in primary operating procedure and secondary operating procedure

the complexities of consistency in the local execution and remote execution and synchronization of the application execution between mobile device and remote server node. However, the replication of intensive components of the mobile application assists in achieving the goals of rich user experiences and offline usability. It means that the application on mobile device still remains functional in offline mode in the situations of unavailability of preconfigured services on the cloud server node. Similarly, in the critical condition application is switched to online mode, however once the resources become available on the mobile device, application can be switched back to the offline mode. The architecture and operation procedure of EECOF are different from the traditional client/server applications. The traditional client/server applications are thin client applications. The client applications provide user agents for interaction with the local computer, whereas the processing logic is implemented on the remote server machines. Examples of such applications include web application, email application, social network applications such as Facebook, and video conferencing applications such as Skype application. In the traditional client/server model, the client component of the application becomes insignificant in the situations of inaccessibility of the server component. Therefore, EECOF framework is attributed with the features of offline usability, on demand access of the preconfigured cloud services and offloading computational load of the local mobile device in the situation of unavailability of sufficient resources for the processing of mobile application on local mobile device.

The following section presents analytical model for the configuration of mobile application on the basis of EECOF framework. The overall intensive mobile application (A) is classified into two distinct categories which are –application on mobile device (A_c) and application on cloud server (A_s). For each category, components of the application are configured on the basis of the computational requirements – either memory size (i.e., (M_c) for A_c and M_s for A_s) or processing length (i.e., P_c for A_c and P_s for A_s). Typically, the application on mobile device is composed of the small intensive and tightly coupled components, whereas the application on cloud server node is composed of big intensive and loosely coupled components of A (Fig. 5).

Let there are n_1 components in A_c and N_2 components in A_s such that $n_1 + n_2 = n$, $A_c \cup A_s = A \in U$ and $A_c \cap A_s = \emptyset$ then by using the particular notations the following expressions are defined with respect to different aspects of EECOF.

For the *application on mobile device*, let T_x^c and T_y^c be the total memory and processing requirements respectively, and q_k and r_k be the respective memory size and processing length of the k^{th} single component $x_k \in A_c$. Then, $Q_c = \{q_k \in N : q_k > 0, 1 \leq k \leq n_1\}$ and $R_c = \{r_k \in R : r_k > 0, 1 \leq k \leq n_1\}$. So, T_x^c and T_y^c are determined as,

$$T_x^c = \sum_{k=1}^{n_1} q_k, \text{ for } q_k \in Q_c \text{ and } |Q_c| \geq 1, \tag{3}$$

and

$$T_y^c = \sum_{k=1}^{n_1} R_k, \text{ for } r_k \in R_c \text{ and } |R_c| \geq 1. \tag{4}$$

In the same way, for the *application on remote server*, if T_x^s and T_y^s represent the total memory and processing requirements respectively, then they can be determined as,

$$T_x^s = \sum_{l=1}^{n_2} q_l, \text{ for } q_l \in X_s \text{ and } |X_s| \geq 1 \tag{5}$$

and

$$T_y^s = \sum_{l=1}^{n_2} R_l, \text{ for } r_l \in R_s \text{ and } |R_s| \geq 1 \tag{6}$$

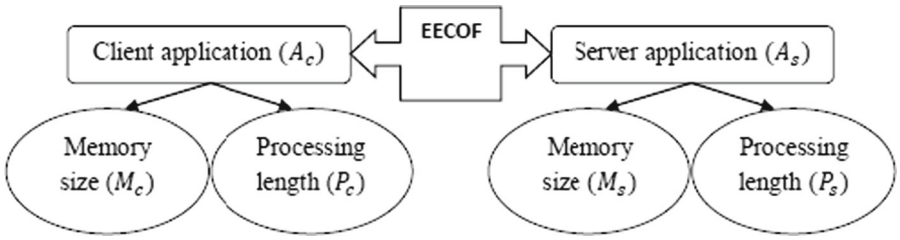

Fig. 5 A typical depiction of EECOF based mobile application

where given that, $Y_s = \{y_l \in N : y_l > 0, 1 \leq l \leq n_2\}$ for the memory size y_l of the l^{th} single component $x_l \in A_s$, and $Z_s = \{z_l \in R : z_l > 0, 1 \leq 1 \leq n_2\}$ for the processing length z_l of the l^{th} single component $x_l \in X_s$.

As $A_c \subset A \in U$, $A_s \subset X \in U$, $A_c \cup A_s = X \in U$ and $A_c \cap A_s = \emptyset$ the memory size of the standalone mobile application is equal to the sum of memory size of application on mobile device and the memory size of application on the remote server. Now the equation (1) can be expressed as,

$$T_x = \sum_{i=1}^{n} q_i = \sum_{k=1}^{n_1} qk + \sum_{l=1}^{n_2} q_l \text{ (as } n = n_1 + n_2).$$

Hence, by using equations (3) and (5), the total memory size of the mobile application can be determined as,

$$T_x = T_x^c + T_x^s \tag{7}$$

This standard equation (7) is useful for validating the total memory size of the mobile application (i.e., $T_x^c = T_x - T_x^s$) and the total memory size of server application (i.e., $T_x^s = T_x - T_x^c$).

Moreover, if α represents the percentage of memory saved by outsourcing intensive components of the application at design time. Then by using (7), α can be calculated as,

$$\alpha = \frac{\tau_x^s}{\tau_x} \times 100. \tag{8}$$

Similarly, by employing equations (4) and (6) the total processing length of the standalone mobile application can be determined by the following standard equation,

$$T_y = T_y^c + T_y^s \tag{9}$$

The equation (9) is useful for validating the total processing length of the application on mobile device (i.e., $T_y^c = T_y - T_y^s$) and the total processing length of server application (i.e., $T_y^s = T_y - T_y^c$).

If β represents the %age of CPU saved by offloading intensive components of the application at design time, then by using (9), β can be easily estimated as follows:

$$\beta = \frac{\tau_y^s}{\tau_y} \times 100. \tag{10}$$

4 Methodology

Experimental Setup We evaluate EECOF by benchmarking the prototype application for the Android operating system platform in the real mobile cloud computing environment. The experimental setup for testing the prototype application in the real wireless mobile network environment is composed of Wi-Fi wireless network of radio type 802.11g, Server node and Samsung Galaxy SII mobile device. We have deployed our own Hybrid Mobile Cloudlet based on OpenStack Cloud Operating System which is used for experimentation and mobile cloud development purposes. The server node is configured in the virtual machine instance deployed in the cloud datacenter. The server node runs Microsoft Windows 7 Professional 32-bit operating system with Intel®core(TM) i5-2500 CPU having 3.3GHz speed and 4.0 GB RAM capacity. The Samsung Smartphone runs Android 4.0.3, dual core ARMv7 Application Processor with 1.2 GHz speed, 16GB memory capacity and 1650mAh battery. Mobile device accesses the wireless network via Wi-Fi wireless network connection of radio type 802.11g, with the available physical layer data rates of 54Mbps. The IaaS model of the cloud datacenter is employed by running Android Virtual Device (AVD) instances of the mobile device on the virtual machine instance of the cloud server node which runs Android 4.1-API Level-17 with ARMv7 Processor having 2389.08 BogoMIPS speed and 1GB RAM capacity.

The client application runs on the mobile device, whereas the server application runs on the remote

cloud server node. The server machine is configured for the provisioning of services to the mobile device in two distinct operating modes of the application. The preconfigured services of the server node utilizes the Software as a Service (SaaS) model of cloud computing for the provisioning of distributed services in the primary operation procedure of EECOF, whereas the Infrastructure as a Service (IaaS) model employed for the provisioning of services in the SOP of mobile application. The virtual device instance is employed on the server machine for the execution of offloaded application at runtime.

We employ the server application by using Microsoft Visual Studio 2010 ASP.NET Web Service Application tool of Visual C#, whereas kSOAP2 API [43] is employed for the configuration of the client application. Java based Android Software development toolkit [40] is deployed for the development of client application. The Android ADB (Android Debug Bridge) Plug in is embedded in the Eclipse application development tool for the development of prototype application. The POP of the client application is implemented by using kSOAP2 library API on the mobile devices for accessing the preconfigured services of the server application. The AVD instance is created on the remote server machine by using Android emulator for the execution of offloaded components of the client mobile application in the SOP. Monitoring tools such as ADB and Dalvik Debug Monitor System (DDMS) [40] are used for the measurement of resources utilization (CPU and RAM), whereas Power Tutor tool [41] is used for the measurement of battery power consumption in distributed application processing.

Prototype Application We develop a prototype application by using Service Oriented Architecture (SOA) of Android applications [40]. The SOA enables to deploy service granularity level for offloading computational load of the mobile application. The prototype application is composed of three computational intensive service components and a single activity component. The service components implement the computational logic of the application, whereas the activity component provides Graphical User Interface (GUI) for interacting with the mobile application. The computational logic of the application is composed of three service components. (a) Sorting service component implements the logic of bubble sort for sorting liner list of integer type values. The sorting logic of the application is tested with 30 different computational intensities (11000-40000). (b) The matrix multiplication service of the application implements the logic of computing the product of 2-D array of integer type values. Matrix multiplication logic of the application is tested with 30 different computational intensities by varying the length of the 2-D array between 160*160 and 450*450. c) The power compute service of the application implements the logic of computing b^e, whereas b is the base and e is the exponent. The power compute logic of the application is tested for 30 different computational intensities by varying the exponent between 1000000 and 200000000. Empirical data are collected by sampling all computational intensities of the application in 30 different experiments and the value of sample mean is signified with 99 % confidence for the sample space of 30 values in each experiment.

Data Gathering and Data Processing The primary data are collected by testing the prototype application in three different scenarios. In the first scenario, all the components of the mobile application are executed on the local mobile device to analyze resources utilization and execution time of the application on the local mobile device. In the second scenario, the intensive components of the mobile application are offloaded at runtime by implementing the traditional computational offloading technique. In this scenario, the resources utilization in distributed processing of the application, data communication over the wireless network medium, and turnaround time of the application on the virtual device instance on the remote server machine are analyzed. In the third scenario, the prototype application is tested for the operating procedures of the proposed framework. We evaluate resources utilization and execution cost of the application in the POP and SOP of the EECOF.

The empirical data are collected for all the computational intensities of every component of the mobile application by executing the component of the mobile application in 30 experiments. Each experiment is conducted 30 times for the evaluation of each parameter to derive the value of point estimator. Data sampling involves the factor of sampling error; hence the sample mean can differ from the population mean. Hence, to signify the goodness of the calculated point estimate; the interval estimate of each sample is

determined. The confidence level of an interval estimate of a parameter indicates the probability that the interval estimate contains the parameter.

5 Results and Discussion

We evaluate the ECC of the components of mobile application in local and remote execution. Experimentation is performed in three different scenarios: (a) execution of mobile application on local mobile device, (b) implementing traditional computational offloading technique [15, 42] for the execution of application and (c) employing EECOF for computational offloading in MCC. Energy efficiency is measured in terms of reduction of energy consumption cost for computational offloading in MCC. The energy efficient nature of EECOF is validated by comparing results of different experimental scenarios. Empirical data are compared from the perspective of local application execution and cloud server based application execution to validate the energy efficiency of proposed framework. The value of ECC for different service components of the prototype application is compared with 30 different computational intensities.

Figure 6 shows difference in ECC of processing sort service component of the application on local mobile device and traditional computational offloading technique for varying computational intensities (list size 11000-40000). The ECC of the sorting service execution is higher in traditional computational offloading as compared to local execution of the service component of the application. The increase in the ECC in traditional computational offloading establishes the fact that additional energy is consumed in offloading the intensive components of the mobile application at runtime. For instance, ECC of executing sorting service on the local mobile device is observed 21.1 J for sorting 11000 values, 37.1 J for sorting 25000 values, 68.6 J for sorting 40000 values. Whereas, the EEC of cloud based processing of sorting service by employing traditional computational offloading is found 49.8 J for sorting 11000 values, 95.4 J for sorting 25000 values and 201.4 J for sorting 40000 values. It shows that by employing traditional computational offloading technique the ECC of sorting operation increases 28.7 J (57.6 %) for sorting list of 11000 values, 58.3 J (61.1 %) for sorting list of 25000 values and 132.8 J (65.9 %) for sorting list of 4000 values. The increase in the ECC is for the reason of runtime component migration. The mechanism of runtime component migration involves the overhead of sending application binary file and data files to the cloud server node and reconfiguration of the delegated application on the cloud server node.

Figure 7 shows difference in ECC of processing sort service component of the application on local mobile device and EECOF based computational offloading. Analysis of the results show that ECC is lower in EECOF based computational offloading as compared local processing of sorting service component of the application. For Instance, in the POP of EECOF the ECC is found 7.4 J for sorting 11000, 14.1 J for sorting 25000 values and 23 J for sorting 40000 values. It shows that by employing the proposed framework, the ECC of sorting operation is reduced in cloud based processing of sorting operation as compared to the execution of sorting operation on mobile device. For instance, the reduction in ECC of sorting operation in EECOF as compared to the execution of service on mobile device is found 14.2 J (65.7 %)

Fig. 6 Difference in ECC of sort service execution on mobile device and traditional computational offloading

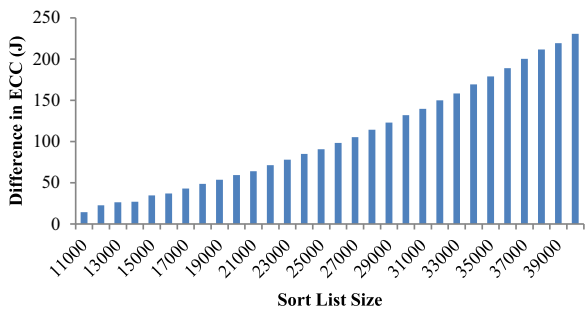

Fig. 7 Difference in ECC of sort service execution on mobile device and EECOF based computational offloading

for sorting 11000 values, 90.6 J (86.5 %) for sorting 25000 values and 230.6 J (90.9 %) for sorting 40000 values. The overall decrease in the ECC in employing EECOF is 90.9 % varying computational intensities of sorting operation (11000-40000).

Figure 8 shows the comparison the ECC in local and cloud based execution of sort service component of the application in different scenarios. It shows that the employment of EECOF for computational offloading reduces the ECC of performing sorting operation on the cloud server node as compared to the ECC of executing sorting service component of the application on local mobile device and traditional computational offloading. The decrease in ECC of sorting operation is for the reason of employing computational task offloading to cloud server node rather than runtime component offloading. The ECC in EECOF based computational offloading reduces 85.1 % for sorting list of 11000 values, 85.2 % for sorting list of 25000 values and 88.6 % for sorting list of 40000 values, as compared to traditional computational offloading. The overall reduction in ECC value for sorting service in EECOF based computational offloading is found is 86(+/−) 0.9 % for varying computational intensities of sorting operation (11000-40000 values).

Similarly, we found that the ECC of the matrix multiplication service execution is higher in traditional computational offloading as compared to local execution of the service component of the application. For instance, the EEC of executing matrix multiplication service on the local mobile device is found 11.5 J for multiplying matrices of 160*160 size, 24.7 J for multiplying matrices of 300*300 size, 69.9 J for multiplying matrices of 450*450 size. Whereas, The EEC of executing matrix multiplication service by employing traditional computational offloading is observed 40 J for multiplying matrices of 160*160 size, 71.9 J for multiplying matrices of 300*300 size, 131.7 J for multiplying matrices of 450*450 size.

Figure 9 shows difference in ECC of processing matrix multiplication service of the application on local mobile device and traditional computational offloading technique for varying computational intensities (matrices size 160*160-450*450). It is found that by employing traditional computational offloading technique the ECC of matrix multiplication operation increases 28.5 J (71 %) for multiplying matrices of 160*160 size, 47.2 J (66 %) for multiplying matrices of 300*300 size and 61.8 J (47 %) for multiplying matrices of 450*450 size. The overall increase in the ECC of matrix multiplication operation is found 47.5 J in traditional computational offloading as compared to the local execution of matrix service for varying intensities (160*160-450*450).

The comparison of ECC of matrix multiplication operation on local mobile device and ECCOF based computational offloading shows that the ECC of the matrix multiplication service execution is lower in EECOF based remote execution as compared to execution of the component of the local mobile device. However, difference in ECC is smaller for lower intensities of the application. For example, percent difference in ECC of matrix multiplication service execution in EECOF based remote processing is up to 0.87 % for intensities between 160*160 and 260*260. Difference in ECC by deploying EECOF for lower intensities of matrix multiplication service is lower for the reason of additional overhead involved (for instance, uploading input parameters and downloading results) in remote processing of application.

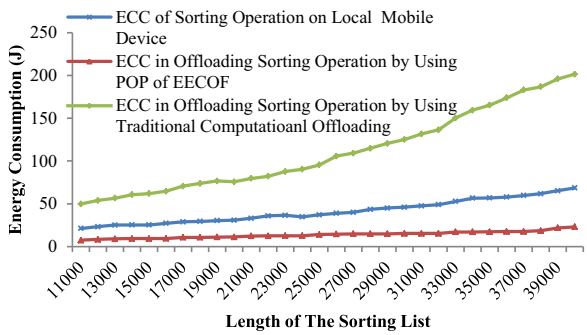

Fig. 8 Comparison of energy consumption cost (ECC) for sorting service in local and cloud based remote execution

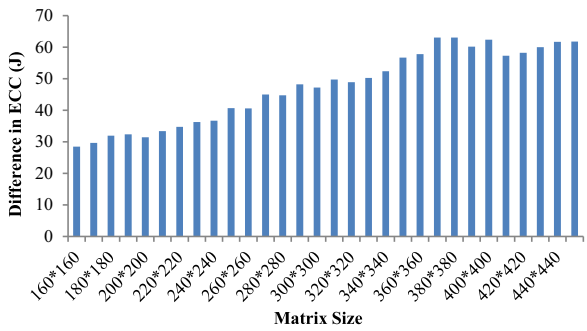

Fig. 9 Difference in ECC of matrix multiplication service execution on mobile device and traditional computational offloading

However, it is found that energy gain is higher in deploying EECOF for higher intensities of the application. For instance, difference in the ECC of executing matrix multiplication service in the POP of EECOF as compared to execution of the application on local mobile device is found up to 4.04 % for higher intensive of matrix multiplication (270*270-450*450). Figure 10 shows difference in ECC of processing matrix multiplication service of the application in the POP of EECOF and the local execution of matrix service for varying higher intensities (270*270-450*450). It shows that deploying EECOF for higher intensities of the application is more significant as compared to low intensities of the application. The overall decrease in the ECC of matrix multiplication operation is found 3(+/-)1.1 J in the POP of EECOF as compared to the local execution of matrix service for varying intensities (160*160-450*450).

Figure 11 shows the comparison the ECC in local and cloud based execution of matrix multiplication service execution in different scenarios. It shows that the employment of EECOF for computational offloading reduces the ECC of performing matrix multiplication operation on the cloud server node as compared to the ECC of executing matrix multiplication service component of the application on local mobile device and traditional computational offloading. The decrease in the ECC of matrix multiplication operation is for the reason of employing computational task offloading to cloud server node rather than runtime component offloading. The reduction ECC of EECOF based computational offloading as compared to traditional computational offloading is found 73 % for multiplying matrices of length 160*160, 70 % for multiplying matrices of length 270*270, 62 % for multiplying matrices of length 350*350 and 50 % for multiplying matrices of length 450*450. The overall reduction in ECC value for matrix multiplication service in POP of EECOF is 64.3 (+/-) 5.4 % with 99 % confidence in the sample space of 30 values.

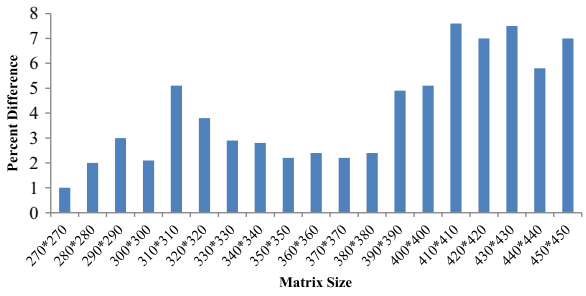

Fig. 10 Difference in energy consumption cost of matrix multiplication service execution on mobile device and EECOF based computational offloading

The ECC of application processing remains lower in EECOF based computational offloading as compared to local execution of the application. For instance, 14.4 J energy is consumed in multiplying 440*440 size matrix on mobile device, where as 10.8 J energy is consumed in deploying POP of EECOF, which shows reduction 25 % in energy consumption while deploying EECOF. Similarly, 15.9 J energy is consumed in multiplying 450*450 size matrix on mobile device, where as 11.3 J energy is consumed in deploying POP of EECOF, which shows reduction 28.9 % in energy consumption while deploying EECOF. However, the total ECC for higher intensities of matrix multiplication operation increases for the reason of energy consumed in downloading the resultant file from the remote server node. For that reason the total ECC in deploying EECOF becomes closer to the ECC of the application on mobile device. However, in comparison to the traditional runtime component migration, the total ECC of EECOF based offloading always remains lower as shown in Fig. 11.

The ECC of the power compute service execution is evaluated by employing the primary operating procedure and secondary operating procedure of EECOF. The employment of POP of EECOF shows that ECC of power compute operation is lower in cloud based processing of the operation as compared to the processing of power compute service on the local mobile device. For instance, in the scenario of local execution of power compute option the ECC is found 2.2 J for computing 2^1000000, 5.4 J for computing 2^60000000 and 67 J for computing 2^2000000000. However, in the POP of EECOF the ECC of power compute operation is found 2 J for computing 2^1000000, 3.5 J for computing 2^60000000 and 11.1 J for computing 2^2000000000. It shows that in the POP of EECOF the ECC reduces 0.2 J (9.1 %) for computing 2^1000000, 1.9 J (35.2 %) for computing 2^60000000 and 55.9 J (83.4 %) for computing 2^ 2000000000. The overall decrease in the ECC of power compute operation in the POP of EECOF is 44 % for 30 varying intensities of the power compute operation (2^1000000-2^2000000000). Figure 12 shows the comparison of ECC for power compute execution on local mobile

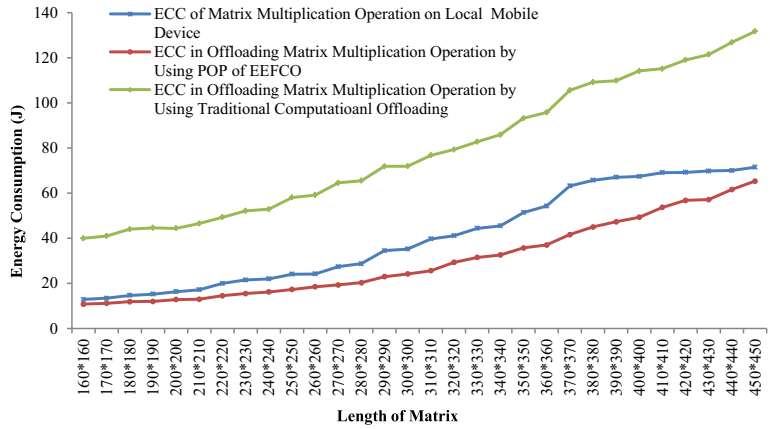

Fig. 11 Comparison of energy consumption cost (ECC) for matrix multiplication service in local and cloud based remote execution

device and cloud server nodes by using POP of EECOF for 30 different computational intensities of the power compute operation.

To evaluate the viability of proposed framework, the power compute service component of the application is also executed by using the SOP of EECOF, wherein the running instance of power compute service component is offloaded to the cloud server node. It is assumed that the power compute service cannot be accessed in the POP of EECOF for the reason of unavailability of preconfigured service on the cloud server node. Mobile device is pushed into the critical condition, wherein the profiler mechanism reports the critical condition and the orchestrator component the framework switches mobile application to the online mode. It determines the power computing task cannot be offloaded by using POP of EECOF; therefore the running instance of power compute application is offloaded to the remote server node. The delegated component is reconfigured on the cloud server node and power compute operation is performed. The significance of active services migration in the SOP of EECOF to cloud server nodes is that it reduces the computational load on SMD which results in saving computing resources (RAM and CPU). However, migration of the active services at runtime involves the additional complications of the establishment and management of distributed platform at runtime. Therefore, the ECC of offloading power compute time is increased in SOP of EECOF. For instance, the ECC in SOP of EECOF as compared to local execution of power compute service increases 59.3 % for computing 2^1000000, 76.7 for computing 2^60000000 and 89.9 % for computing 2^2000000000.

Analysis of the results shows lower energy consumption cost by employing ECCOF for cloud based processing for the intensive component of mobile application. It is observed that the additional cost of application binary code migration and active data state migration to the cloud server node is reduced

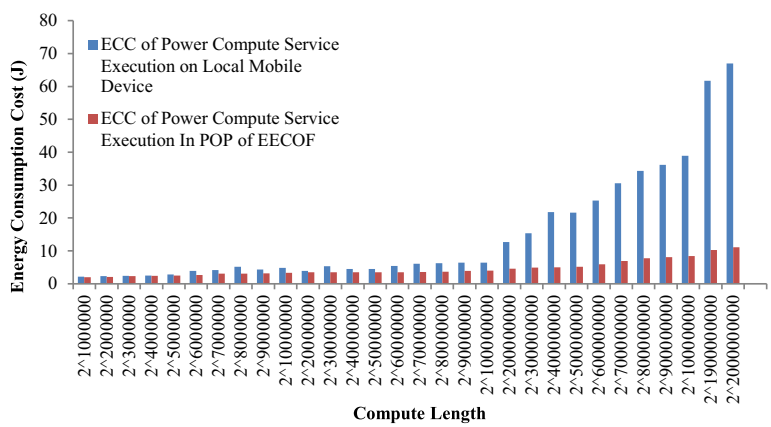

Fig. 12 Comparison of energy consumption cost for power compute service execution on local mobile device and cloud server node in the POP of EECOF

by employing EECOF for cloud based processing of computationally intensive mobile applications. As a result, the energy consumption cost of the component of the mobile application is reduced. For instance, by employing EECOF the size of data transmission for sorting service is reduced 74.8 %, and the energy consumption cost is reduced 86.7 % compared to the traditional computational offloading technique. Similarly, the size of data transmission for matrix multiplication operation is reduced 92.8 % and energy consumption cost is reduced 64.2 % compared to the traditional computational offloading technique. Hence, EECOF provides a lightweight and energy efficient mechanism for computational offloading in MCC.

6 Conclusion and Future Work

Traditional COFs are based on the establishment of distributed platform at runtime and lack of distributed architecture for the intensive mobile application as a result additional energy is consumed in component offloading at runtime. Hence, EECOF addresses the issue of additional energy consumption in computational offloading for MCC by deploying a distributed architecture and employs lightweight procedure for minimizing the overhead of runtime component offloading. EECOF leverages the SaaS model for the configuration of intensive components on the cloud server node and the IaaS model for adaptive offloading of mobile application. The incorporation of preconfigured services access technique and runtime computational offloading technique facilitates in the optimal deployment procedure with minimal energy consumption cost for the establishment of distributed platform in MCC. The dual operating nature of the proposed framework contributes to the versatility and robustness of the distributed and elastic model for intensive mobile application in MCC.

The additional cost of application binary code migration and active data state migration to the cloud server node is reduced by employing EECOF for cloud based application processing. As a result, the ECC of the component of the mobile application is reduced. Analysis indicates that by employing EECOF the size of data transmission over the wireless network medium is reduced up to 84 % and energy consumption cost is reduced up to 69.9 % by offloading different components of the prototype application. Hence, EECOF provides an energy efficient application layer solution for computational offloading in MCC. The future research work includes extending the scope this research to address the issues of consistency of simultaneous application execution between local mobile device and remote cloud server node, and seamless application execution in cloud based application processing of intensive mobile application.

Acknowledgment This research is carried out as part of the Mobile Cloud Computing research project funded by the Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant with reference UM.C/HIR/MOHE/FCSIT/03.

References

1. Holman, R.: Mobile Cloud Computing: $9.5 billion by 2014. http://www.juniperresearch.com/analyst-xpress-blog/2010/01/26/mobile-cloud-application-revenues-to-hit-95-billion-by-2014-driven-by-converged-mobile-services/ (2010). Accessed on 18 August 2013
2. Prosper Mobile Insights, Smartphone/tablet user survey. http://prospermobileinsights.com/Default.aspx?pg=19 (2011). Accessed on 20 July 2013
3. Albanesius, C.: Smartphone shipments surpass PC shipments for first time. What's next? http://www.pcmag.com/article2/ Accessed on 15 December 2013
4. Shiraz, M., Gani, A., Khokhar, H.R., Buyya, R.: A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. IEEE Commun. Surv. Tutorials **15**(3), 1294–1313 (2013). doi:10.1109/SURV.2012.111412.00045
5. Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., Buyya, R.: Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open issues. IEEE Commun. Surv. Tutorials **16**(1), 337–368 (2014)
6. Rings, T., Caryer, G., Gallop, J., Grabowski, J., Kovacikova, T., Schulz, S., Stokes-Rees, I.: Grid and cloud computing: Opportunities for integration with the next generation network. J. Grid Comput. **2009**(7), 375–393 (2009). doi:10.1007/s10723-009-9132-5
7. Rimal, P.B., Jukan, A., Katsaros, D., Goeleven, Y.: Architectural requirements for cloud computing systems: An enterprise cloud approach. J. Grid Comput. **2011**(9), 3–26 (2011). doi:10.1007/s10723-010-9171-y
8. Chohan, N., Bunch, C., Krintz, C., Canumalla, N.: Cloud platform datastore support. J. Grid Comput. **2013**(11), 63–81 (2013). doi:10.1007/s10723-012-9238-z
9. Shamsi, J., Ali, K.M., Qasmi, A.M.: Data-intensive cloud computing: Requirements, expectations, challenges, and solutions. J. Grid Comput. **2013**(11), 281–310 (2013). doi:10.1007/s10723-013-9255-6
10. Troger, P., Merzky, A.: Towards standardized job submission and control in infrastructure clouds. J. Grid Comput. **2014**(12), 111–125 (2014)

11. Shiraz, M., Ahmed, E., Gani, A., Han, Q.: Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing. J. Supercomput. **67**(1), 84–103 (2014). doi:10.1007/s11227-013-0988-6
12. Amazon S3 [Online Available] http://status.aws.amazon.com/s3-20080720.html (Accessed on 20th July 2011)
13. Cuervo, E., Balasubramanian, A., ChoK, D.O., Wolman, A., Saroiu, S., Chandra, R., Bahlx, P.: MAUI: Making Smartphones Last Longer with Code Offload MobiSys'10. San Francisco (2010)
14. Zhang, X., Kunjithapatham, A., Jeong, S., Gibbs, S.: Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. Mobile Netw. Appl. **16**(3), 270–285 (2011)
15. Hung, H.S., Shih, S.C., Shieh, P.J., Lee, P.C., Huang, H.Y.: Executing mobile applications on the cloud: Framework and issues. Comput. Math. Appl. **63**(2), 573–587 (2012)
16. Messer, Greenberg, I., Bernadat, P., Milojicic, D., Chen, D., Giuli, T.J., Gu, X.: Gu Towards a Distributed Platform for Resource-Constrained Devices. Hewlett-Packard Company (2002)
17. Giurgiu, Riva, O., Juric, D., Krivulev, I., Alonso, G.: Calling the cloud: Enabling mobile phones as interfaces to cloud applications. Middleware'09 Proceedings of the ACM/IFIP/USENIX 10th International Conference on Middleware, pp. 83–102. ACM Press (2009)
18. Chun, B.G., Maniatis, P.: Augmented Smartphone Applications Through Clone Cloud Execution. Intel Research Berkeley (2009)
19. Kovachev, D., Klamma, R.: Framework for computation offloading in mobile cloud computing. Int. J. Interact. Multimedia Artif. Intell. **1**(7), 6–15 (2012)
20. Apple – iCloud http://www.apple.com/icloud/ Accessed on 1 January 2013
21. Introducing Amazon Silk. http://amazonsilk.wordpress.com/2011/09/28/introducing-amazon-silk/ Accessed on 1 January 2013
22. Yang, L., Cao, J., Cheng, H.: Resource Constrained Multi-user Computation Partitioning for Interactive Mobile Cloud Applications. Hong Kong Poly-technical University, Technical Report (2012)
23. Abebe, E., Ryan, C.: Adaptive application offloading using distributed abstract class graphs in mobile environments. J. Syst. Softw. **85**(12), 2755–2769 (2012)
24. Goyal, S., Carter, J.: A Lightweight Secure Cyber Foraging Infrastructure for Resource-Constrained Devices WMCSA 2004 6th IEEE Workshop. IEEE (2004)
25. Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE Pers. Commun. **8**(4), 10–17 (2001)
26. Oh, Lee, S., Lee, E.: An adaptive mobile system using mobile grid computing in wireless network. In: Proceedings of the 6th International Conference on Computational Science and Its Applications (ICCSA 2006), pp. 49–57. Glasgow, UK (2006)
27. Chunlin, Layuan, L.: Energy constrained resource allocation optimization for mobile grids. J. Parallel Distrib. Comput. **70**(3), 245–258 (2010)
28. Begum, Y., Mohamed, M.: A DHT-based process migration policy for mobile clusters. In: 7th International Conference on Information Technology, pp. 934–938. Las Vegas (2010)
29. Tilevich, E., Smaragdakis, Y.: J-orchestra: Automatic java application partitioning. ECOOP 2002—Object-Oriented Programming, pp. 178-204 (2006)
30. Musunoori, B.S., Horn, G.: Intelligent ant-based solution to the application service partitioning problem in a grid environment. In: 6th International Conference on Intelligent Systems Design and Applications, ISDA'06, pp. 416–424 (2006)
31. Newton, R., Toledo, S., Girod, L., Balakrishnan, H., Madden, S.: Wishbone: Profile-based partitioning for sensornet applications. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI), pp. 395–408. Boston (2009)
32. Bi-Ru, D., Lin, C.I.: Efficient Map/Reduce-Based DBSCAN Algorithm with Optimized Data Partition. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference, pp. 59–66. Honolulu (2012)
33. Gu, X., Nahrstedt, K., Messer, A., Greenberg, I., Milojicic, D.: Adaptive offloading inference for delivering applications in pervasive computing environments. In: Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003), pp. 107–114 (2003)
34. Chu, H., Song, H., Wong, C., Kurakake, S., Katagiri, M.: Roam, a seamless application framework. J. Syst. Softw. **69**(3), 209–226 (2004)
35. Satyanarayanan, M., Bahl, P., Caceres, R.: The Case for VM-Based Cloudlets in Mobile Computing IEEE Computing Society (2009)
36. Dou, Kalogeraki, V., Gunopulos, D., Mielikainen, T., Tuulos, V.H.: Misco: A MapReduce Framework for Mobile Systems, PETRA'10 Samos. ACM Press, Greece (2010)
37. Chun, G., Ihm, S., Maniatis, P., Naik, M., Patti, A.: CloneCloud: Elastic Execution between Mobile Device and Cloud, EuroSys'11. ACM Press, Salzburg Austria (2011)
38. Kumar, K., Lu, H.Y.: Cloud computing for mobile users: Can offloading computation save energy. Comput. IEEE Comput. Soc. **43**(4), 51–56 (2010)
39. Shiraz, M., Gani, A., Khokar, R.H.: An Extendable Simulation Framework for Modeling Application Processing Potentials of Smart Mobile Devices for Mobile Cloud Computing, Proceedings of Frontiers of Information Technology 2012. Pakistan (2012)
40. Android Developers. http://developer.android.com/index.html Accessed on 10 July 2011
41. Power Tutor. http://ziyang.eecs.umich.edu/projects/powertutor/ Accessed on 15 April 2012
42. Shiraz, M., Gani, A.: A lightweight active service migration framework for computational offloading in mobile cloud computing. J. Supercomput. **68**(2), 978–995 (2014). doi:10.1016/j.jnca.2014.04.009
43. Ksoap2-android. https://code.google.com/p/ksoap2-android/ Accessed on 1 May 2013