Wireless Pers Commun (2017) 95:4403–4423
DOI 10.1007/s11277-017-4086-6

A Distributed and Elastic Application Processing Model for Mobile Cloud Computing

Muhammad Shiraz[1] · Abdullah Gani[1] · Rashid Khokhar[2] · Azizur Rahman[2] · Mohsin Iftikhar[2,3] · Naveen Chilamkurti[3]

Published online: 13 March 2017

Abstract The latest developments in mobile computing technology have increased the computing capabilities of smart mobile devices (SMDs). However, SMDs are still constrained by low bandwidth, processing potential, storage capacity, and battery lifetime. To overcome these problems, the rich resources and powerful computational cloud is tapped for enabling intensive applications on SMDs. In Mobile Cloud Computing (MCC), application processing services of computational clouds are leveraged for alleviating resource limitations in SMDs. The particular deficiency of distributed architecture and runtime partitioning of the elastic mobile application are the challenging aspects of current offloading models. To address these issues of traditional models for computational offloading in MCC, this paper proposes a novel distributed and elastic applications processing (DEAP) model for intensive applications in MCC. We present an analytical model to evaluate the proposed DEAP model, and test a prototype application in the real MCC environment to demonstrate the usefulness of DEAP model. Computational offloading using the DEAP model minimizes resources utilization on SMD in the distributed processing of intensive mobile applications. Evaluation indicates a reduction of 74.6% in the overhead of runtime application partitioning and a 66.6% reduction in the CPU utilization for the execution of the application on SMD.

Keywords Application processing model · Mobile Cloud Computing · Smart mobile devices · Distributed application

✉ Mohsin Iftikhar
miftikhar@csu.edu.au

[1] Center for Mobile Cloud Computing Research (C4MCCR), Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia

[2] School of Computing and Mathematics, Charles Sturt University, Wagga Wagga, NSW 2678, Australia

[3] Department of Computer Science and IT, School of Engineering and Mathematics, La Trobe University, Melbourne, Australia

1 Introduction

Technological enhancements have enriched mobile devices with the attributes of autonomous operating systems and customized user application support. SMDs are high featured and multifunctional computing and communication devices which have become universal replacements for Personal Digital Assistants (PDA's). SMDs incorporate the computing capabilities of a handheld computer and the communication capabilities of an ordinary cellular phone by providing multimodal connectivity and user applications support [1]. SMDs are predicted to become the primary computing devices for various computational intensive applications [2]. Examples of such applications include natural language translators [3, 4], speech recognizers [5, 6], optical character recognizers, image processors [7, 8], online computational intensive games, video processing [9] and wearable devices for patients [10]. However, such applications require higher computing power, storage capacity, and battery lifetime on SMDs [11]. However, SMDs currently have limitations in memory capacity, processing potential, battery lifetime and wireless network access medium.

MCC is a new advancement which aims at extending cloud services to SMDs for augmenting the computing potentials of resources constraint SMDs [12, 13]. MCC addresses the issue of resources limitations in SMDs by employing resources and services of computational clouds.

Successful practices of cloud computing for stationary computers motivate to leverage cloud resources and services for SMDs. Cloud computing employs different service provision models for the provisioning of cloud resources and services to SMDs [14]. Several online file storage services are available on cloud server for augmenting the storage potential of client devices, such as Amazon S3 [15], Google Docs [16], MobileMe [17], and Drobox [18].

Applications which are partitioned at different granularity level into independent components are called elastic applications. Application partitioning is employed either statically or dynamically [21] for enabling intensive applications on SMDs. However, a number of issues are raised [2, 3] in the development, deployment and management of current cloud based distributed application deployment frameworks for intensive applications which obstruct optimization goals of distributed application processing in MCC [19–22]. Current offloading frameworks lack in the deployment of distributed architecture for intensive mobile application, which results in the runtime partitioning of mobile applications for the establishment of ad-hoc distributed processing platform for application processing. However, runtime partitioning of elastic mobile applications involves additional computing resources utilization in implementing application profiling and solving [23]. The deficiency of distributed architecture and runtime partitioning of the application for outsourcing intensive components of the application make the distributed platform resources intensive and time consuming. However, the limited resources nature of SMDs necessitates lightweight procedures for the distributed deployment of intensive mobile applications which require minimal resources utilization in outsourcing computational load to cloud server node [21].

This paper proposes a Distributed and Elastic Application Processing (DEAP) model which addresses the issues of traditional application offloading frameworks [24–26]. The availability of centralized resources and centralized monitoring in cloud datacenters motivates the need for explicitly configured client/server applications as an alternative for complex and resources starving runtime application offloading of intensive mobile

applications [20]. DEAP incorporates distributed architecture for separating the intensive components of mobile application at design time, which reduces the instances of runtime application partitioning and therefore minimizes the additional overhead of runtime partitioning of the application. The graphical and mathematical models of DEAP are presented.

We formulate the performance of the elasticity attributes of DEAP by modeling the computational resources utilization in runtime application profiling and optimization. The performance gains of DEAP are evaluated by quantitative analysis of the intensive mobile application using the mathematical model. Analysis of the results shows the optimal and distributed nature of DEAP model for the intensive application deployment for MCC. Intensive mobile applications which are distributed by design and elastic by nature are promising alternatives of current offloading frameworks for intensive mobile applications. The development of distributed applications on the basis of DEAP framework results in substantial performance gains and enhancement in overall performance of application development, deployment and processing in MCC. The paper is organized as follows.

Section 2, discusses traditional application offloading frameworks. Section 3 presents the DEAP model in graphical model and mathematical form to clearly explain the idea of the proposed solution. Section 4 evaluates DEAP model through a quantitative analysis. Finally, simulation results are given in Sect. 5 followed by conclusions and future directives in Sect. 6.

2 Related Work

A number of related distributed and elastic application processing models are proposed for offloading intensive applications such as: (a) decentralized virtual cloud computing environment for mobile devices [27], (b) local surrogate based distributed computing platform [28], (c) centralized cloud computing environment for mobile devices [29], and (d) centralized cloud computing datacenters based cloud computing environment [26]. Current computational offloading frameworks employ diverse technique for the establishment of runtime distributed application execution platform. For example, virtual machine (VM) migration based Cloudclone framework [30] seamlessly offloads cloned image of the running applications on SMD to the nearby computer. The framework exploits various augmentation strategies for different types of applications and reduces the dynamic transmission overhead of application code by deploying a simple approach for synchronization. Similarly, VM based Cloudlets architecture [22] is deployed for process offloading. The framework exploits the tactic of copying the entire processing environment of the mobile device into remote cloudlet. A cloudlet is a trustable remote computer which provides the services of outsourced processing of application to SMD. A VM based CloneCloud framework [31] is based on partitioning of the application on thread basis. The framework is based on cloning mobile device application processing environment on to remote host which involves the issues of privacy and access control. Mirror server [32] augments SMDs by providing different types of services; security (file scanning), storage (file caching) and computation offloading. A mirror server is a powerful server configured in Telecommunication Server Provider (TSP) that maintains VM template for each of the different types of mobile devices platform. The VM template for each mobile device is kept with default company settings and a new VM instance is created for offloaded component of the mobile application. Cloud datacenter based framework [33] employs

virtual machine based application offloading procedure. The framework deploys application level process migration on the Android platform. Fresh VM instance is created on the cloud server and state of the application is cloned to the newly created VM instance on the cloud server node.

Similarly, AIDE [34] is a dynamic distributed framework for resources constrained devices. An application profiling component establishes the feasibility of offloading. The partitioning component of the AIDE partitions the application dynamically by following a partitioning policy. In [35] a middleware framework is proposed for sharing the application processing dynamically between cloud server and mobile client. The framework implements both static partitioning and dynamic partitioning strategies. MAUI [25] uses dynamic application profiling and partitioning approach to maximize energy saving for mobile devices. The framework is based upon method state migration instead of method code migration. MAUI involves the overhead of application profiling, dynamic partitioning, migration, and reintegration on mobile device which requires computing resources abundantly. MAUI lacks of supporting remotely executing virtualized methods calling native functions and requires programmers to annotate methods as REMOTABLE, which is an additional effort from the development perspective. Similarly, elastic application model [26] is a middleware framework for elastic mobile applications. The framework is based upon dynamic distributed processing platform at application level. Weblets are deployed for dynamic partitioning and migration at runtime. The critical aspects are that the framework requires extensive overhead of application profiling, dynamic runtime partitioning, migration, reintegration, and rigorous synchronization on mobile devices for offload processing.

Existing computational offloading frameworks for MCC [24–26] are the analogous extensions of pervasive computing [27] models or local distributed [28] application execution models for distributed application processing. Therefore, current offloading frameworks are deficient in the deployment of distributed system architecture for the development and deployment of applications for MCC. Current frameworks focus on the establishment of runtime distributed platform which utilizes additional resources in the deployment and management of distributed application processing platform.

Computing resources of the SMDs are utilized in arbitration with cloud servers for the selection of remote server node, dynamic application profiling, runtime solving of critical condition, application migration and reintegration and rigorous synchronization with cloud servers for the entire duration of distributed platform [23]. Consequently, current distributed application deployment models employ heavyweight procedures in leveraging application processing services of computational clouds for SMDs. The mobile nature, compact design, limited computing potential and wireless medium attributes of SMDs necessitate optimal and lightweight procedures for distributed application deployment in MCC.

3 Proposed Distributed and Elastic Application Processing (DEAP) Model

We propose DEAP model for enabling computationally intensive applications on SMDs. DEAP is distributed from the development and deployment perspectives and elastic in nature. Distributed features include explicitly defined distributed architecture for the deployment of distributed processing environment. Whereas, considering the importance,

versatility and robustness of current elastic application frameworks [25, 26], DEAP is attributed with the elasticity features. The attributes of elastic mobile applications include ad-hoc platform creation, partitioning of intensive components, adaptive offloading, and transparency in distributed application deployment [30]. DEAP is based on the design time separation of the loosely coupled and tightly coherent components (as represented in set S in Sect. 4) and tightly coupled and loosely coherent components (as represented in set C in Sect. 4) of the mobile application and incorporation of the elasticity attributes for dynamic processing management on SMD.

The centralized management and availability of services in cloud datacenters motivate for employing explicitly designed client and server components of the intensive mobile applications for MCC. Therefore, DEAP model is based on explicitly defined client and server components of the intensive mobile applications. DEAP addresses the issue of deficiency of distributed architecture in remote processing of intensive components of mobile applications. DEAP is designed with two objectives: (1) Incorporation of standardized distributed architecture for the design, development and implementation of intensive applications in MCC. (2) Deployment of elasticity attributes for coping with dynamic application processing load on SMDs. DEAP employs two tiered architecture by explicitly defining the client and server components of the intensive mobile application at design time. DEAP Client is composed of location aware, tightly coupled or slight intensive components of the mobile application. DEAP Server is composed of maximum possible loosely coupled, tightly cohesive, location unaware and intensive components of the mobile application.

The client component of DEAP architecture is composed of SMD synchronizer, distributed middleware, profiler, optimizer, migrator and mobile node manager. DEAP Client is designed with the objective to include location aware, slight intensive or tightly coupled components of the intensive mobile application. DEAP Client utilizes the services of explicitly defined DEAP Server by using inter-process communication (IPC) mechanism. Traditionally, a large number of distributed applications are deployed over the internet such as Email, Facebook and Web application which provide thin client framework for client devices.

Client's devices provide the user interface and the processing logic and implement on the server component of the application. However, DEAP Client is distinct from traditional thin clients for the reason of the enrichment of smartness of services on client component of the application. DEAP is distributed for the reason of having explicitly defined client and server components and elastic for the reason of runtime component offloading to cloud server node.

Considering the mobile nature and intrinsic limitations of wireless medium, DEAP proposes to process the slight intensive or tightly coupled components of the mobile application on SMD which contributes to the richness and smartness of local services and offline usability of the mobile application. DEAP proposes two independent operating procedures for the implementation of distributed platform of intensive mobile applications in MCC: Primary Operating Procedure (POP) and Secondary Operating Procedure (SOP).

DEAP Client is capable to implement processing logic on SMD for enhancing the smartness of local services on SMD. Figure 1 shows the sequence diagram for POP of DEAP client. In the POP of DEAP client application invokes the preconfigured services on the server application running on the cloud server node. Once the execution of remotely invoked services is completed, results are returned to the client application running on mobile device. The synchronizer component of DEAP model enables synchronization between application running on the mobile device and cloud server node. The primary

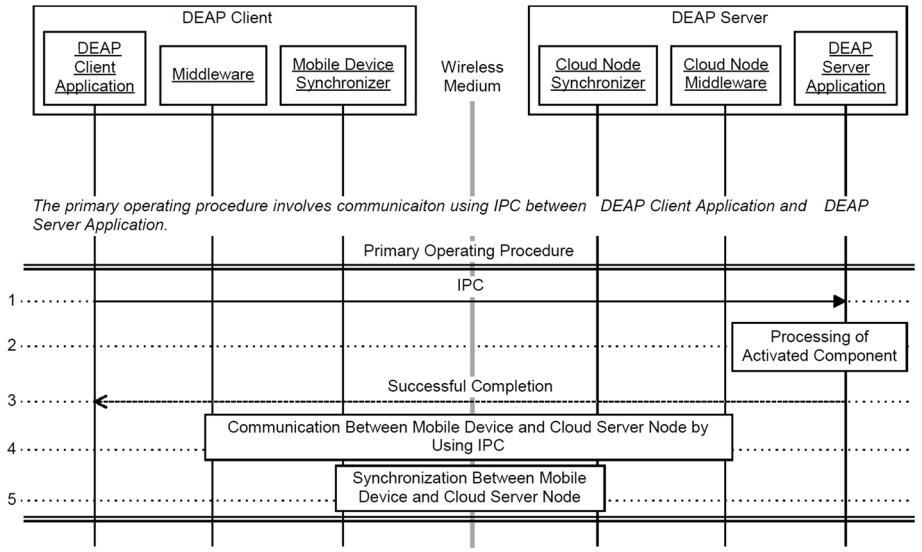

Fig. 1 Sequence diagram for primary operating procedure of DEAP model

communication mechanism in the POP is to invoke the processing of remote services by using IPC mechanism rather than send and receive messages. However, whenever data transmission is required between DEAP client application and DEAP server application, appropriate logic can be implemented for sending and receiving data. The POP of DEAP client is a simple and optimal procedure for remote processing of intensive components of the mobile application. POP utilizes the significance of explicitly designed DEAP server running on cloud node and does not involve the overhead of runtime optimization such as profiling, optimization and migration.

Figure 2 shows sequence diagram for the SOP of DEAP framework. The SOP of DEAP is employed for coping with the dynamic application processing load on the SMDs. DEAP client employs SOP at times when the computing requirements of the client application cannot be fulfilled on SMD. In order to cope with such critical situations, DEAP client activates the Profiler mechanism to identify the larger intensive components of the application; whereas, the Optimizer component resolves the problem of critical situation on the basis of input provided by Profiler. Similarly, Migrator component arbitrates with cloud servers for the migration of intensive components of the DEAP client application at runtime. It is important to highlight that the SOP does not utilize the services of explicitly defined DEAP server; instead, the migrator component arbitrates with cloud servers to facilitate remote execution services on casual basis.

Considering the availability of centralized resources and management of centralized datacenters in computational clouds, explicitly configured DEAP server is an appropriate alternative for ad-hoc surrogates based remote servers of traditional offloading frameworks [27, 28]. Figure 3 shows the architecture of DEAP model. DEAP Server is deployed on cloud server node which eliminates the overhead of arbitration for the selection of appropriate remote host and migration of intensive partitions of the mobile application. The server component of the DEAP architecture is composed of cloud synchronizer and distributed middleware and cloud server node manager. The services of DEAP server are accessible to DEAP Client in POP of DEAP framework. DEAP server utilizes the services

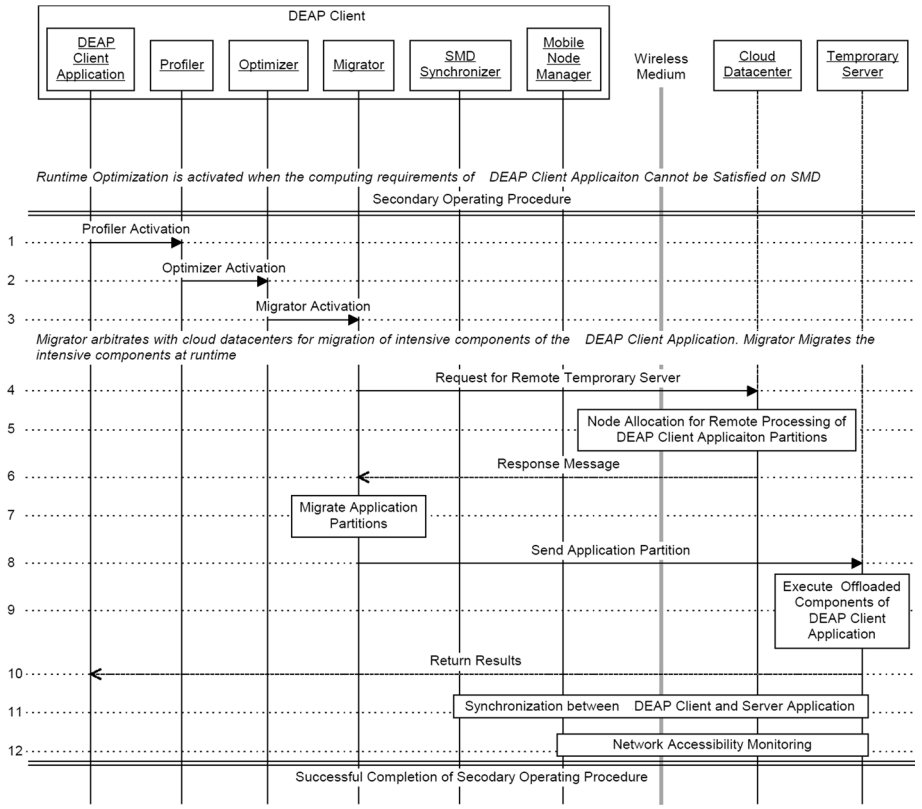

Fig. 2 Sequence diagram for secondary operating procedure of DEAP model

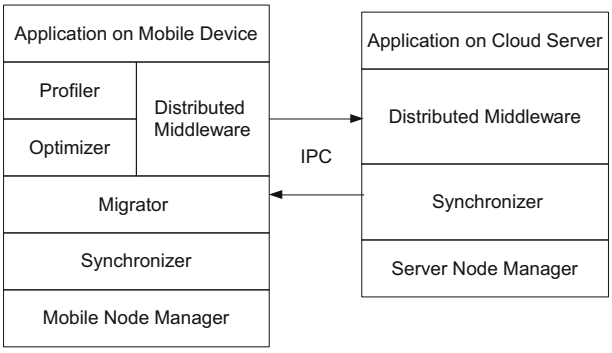

Fig. 3 Architecture of DEAP model

of distributed middleware for communication with DEAP client. The Synchronizer component on both SMD and cloud server is responsible for the synchronization between DEAP client application and DEAP server application. Synchronizer provides different types of services in the POP and SOP. In POP synchronizer coordinates for the synchronization between DEAP client application and DEAP server application. Whereas, the

intensive components of the application are offloaded to cloud server node by employing application level partitioning in the same way as traditional offloading frameworks [24–26].

The offloaded components of the application are executed on temporarily created server node of cloud datacenters. In such scenario the role of synchronizer is to coordinate between elastic DEAP client mobile application and remote offloaded components of the application. The responsibility of the Synchronizer is to ensure the consistency of transmissions between DEAP client and DEAP sever in the POP and DEAP client application and temporarily allocated cloud server in the SOP. Mobile node manager and Server manager are responsible for communication between SMD and cloud datacenters. Mobile node manager copes with the challenges of mobility in the wireless environment and cloud server manager is responsible for ensuring the provision of cloud services to SMD.

4 Analytical Model to Evaluate DEAP Model

In this section we present an analytical model to evaluate traditional elastic mobile applications and DEAP framework. We model the overhead of runtime partitioning by formulating the computational resources utilization in application profiling and partitioning.

Mobile applications have distinct framework which is different from the framework of traditional applications for personal computers. However, application frameworks differ for different mobile operating systems [36]. For instance, applications for Android platform are composed of different components such as activity, services, content provider and broadcast receiver [37]. These application components are the fundamental building blocks of Android applications. Each component has a different point through which the system can enter the application. Each component is a unique building block that helps to define the overall behavior of the application. The following notations are used in the proposed analytical model for DEAP framework.

Let X be an intensive mobile application consists of $n \in N$ operations or components such that $X \in U$ and $|X| = n$, where U represents the universal set and N represents the set of natural numbers. Let $x_i (i = 1, 2, \ldots, n)$ be any single component in X, then x_i can be defined as $\{x_i \in U \ : \forall i \in N, |U| \geq 1\}$ for $X = \{x_1, x_2, \ldots, x_n\}$.

The computational intensity of the application comprises the sum of computational requirements of all the components in X. CloudSim [38] is a simulation toolkit for modeling such a problem in cloud computing environments and evaluation of resource provisioning algorithms. This study adopts a similar means to model the memory (RAM) requirement and processing intensity i.e., amount of CPU requirement for millions of instructions (MI) to accomplish the application.

L and T_p be the total memory and processing requirements respectively, and Y and Z be the corresponding sets of memory size and processing length for X. Then, $Y = \{y_i \in N : y_i > 0, \forall i\}$ when is the memory (RAM) size of the ith single component $x_i (\forall i)$. And the total memory requirement of the application can be defined as,

$$T_m = \sum_{i=1}^{n} y_i, \quad \text{for} \quad y_i \in Y \quad \text{and} \quad |Y| \geq 1. \tag{1}$$

Similarly, if z_i represents the processing length (amount of CPU) of the ith component x_i, then $Z = \{z_i \in R : z_i > 0, \forall i\}$. when R represents the set of real numbers. Hence the total processing requirement of the application can be defined as,

$$T_p = \sum_{i=1}^{n} z_i, \quad \text{for} \quad z_i \in Z \quad \text{and} \quad |Z| \geq 1. \tag{2}$$

4.1 DEAP Model Based Mobile Application

AP model is based on the design time separation of the components of the application. A simplified depiction of the DEAP model based intensive mobile application is presented in Fig. 4. In DEAP model, the overall intensive mobile application (X) is classified into two defined categories which are—client application (X_c) and server application (X_s). For each category, the model organises the components of the application on the basis of the computational requirements—either memory size (i.e.,Y_c for X_c and Y_s for X_s) or processing length (i.e., Z_c for X_c and Z_s for X_s). Typically, the client application is composed of the small intensive and tightly coupled components, whereas the server application is composed of big intensive and loosely coupled components of X.

Let there are n_1 components in X_c and n_2 components in X_s such that $n_1 + n_2 = n$, $X_c \cup X_s = X \in U$ and $X_c \cap X_s = \emptyset$, then by using the respective notations the following expressions can be easily defined with respect to different aspects of DEAP model.

For the DEAP client mobile application, let T_m^c and T_p^c be the total memory and processing requirements respectively, and y_k and z_k be the respective memory size and processing length of the kth single component $x_k \in X_c$. Then, $Y_c = \{y_k \in N : y_k > 0, 1 \leq k \leq n_1\}$ and $Z_c = \{z_k \in R : z_k > 0, 1 \leq k \leq n_1\}$. So, T_m^c and T_p^c are determined as,

$$T_m^c = \sum_{k=1}^{n_1} y_k, \quad \text{for} \quad y_k \in Y_c \quad \text{and} \quad |Y_c| \geq 1, \tag{3}$$

and

$$T_p^c = \sum_{k=1}^{n_1} z_k, \ \text{for} \quad z_k \in Z_c \quad \text{and} \quad |Z_c| \geq 1. \tag{4}$$

Similarly for the DEAP server mobile application, if T_m^s and T_p^s represent the total memory and processing requirements respectively, then they can be determined as,

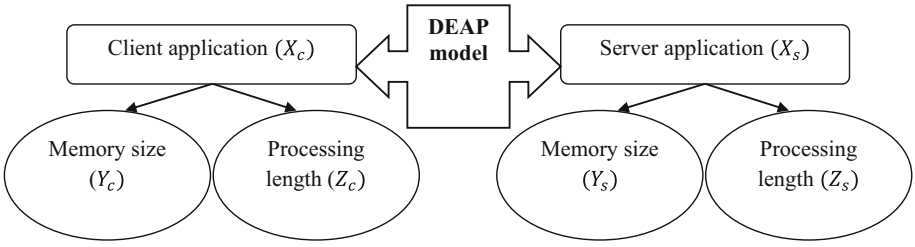

Fig. 4 A typical depiction of DEAP model based mobile application

$$T_m^s = \sum_{l=1}^{n_2} y_l, \quad \text{for} \quad y_l \in Y_s \quad \text{and} \quad |Y_s| \geq 1 \tag{5}$$

and

$$T_p^s = \sum_{l=1}^{n_2} z_l, \quad \text{for} \quad z_l \in Z_s \quad \text{and} \quad |Z_s| \geq 1. \tag{6}$$

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 lth single component $x_l \in X_s$, and $Z_s = \{z_l \in R : z_l > 0, 1 \leq l \leq n_2\}$ for the processing length z_l of the lth single component $x_l \in X_s$.

As $X_c \subset X \in U$, $X_s \subset X \in U$, $X_c \cup X_s = X \in U$ and $X_c \cap X_s = \emptyset$, the memory size of the standalone mobile application is equal to the sum of memory size of client application and the memory size of server application. Now the Eq. (1) can be expressed as,

$$T_m = \sum_{i=1}^{n} y_i = \sum_{k=1}^{n_1} y_k + \sum_{l=1}^{n_2} y_l (asn = n_1 + n_2).$$

Hence, by employing Eqs. (3) and (5), the overall total memory size of the standalone intensive mobile application is determined as,

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

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

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

$$u = \frac{T_m^s}{T_m} \times 100. \tag{8}$$

Similarly, by using Eqs. (2), (4) and (6) the overall total processing length of the standalone mobile application can be determined by the following standard equation,

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

The Eq. (9) is helpful for validating the total processing length of client application (i.e., $T_p^c = T_p - T_p^s$) and the total processing length of server application (i.e., $T_p^s = T_p - T_p^c$).

Now, if v represents the percentage of CPU saved by outsourcing intensive components of the application at design time. Then by using (9), v can be easily estimated as,

$$v = \frac{T_p^s}{T_p} \times 100. \tag{10}$$

4.2 Modeling Profiler Overhead

The elasticity aspect of DEAP framework employs runtime component offloading for coping with the dynamic processing loads on SMD. However, runtime component offloading is implemented as the secondary operation procedure. Investigation of the

application processing on Android devices indicates that the resources utilization of application profiling depends on the number of components in the mobile application and the computational intensity of the components of the mobile application [23]. It means that profiling larger intensive component of the mobile application results in higher resources utilization for a longer period of time. The overhead of the single instance of profiler activation can be modeled on the basis of the number of components in the mobile application with a runtime profiling rate.

Let a_i be the overhead of profiler activation at runtime of the ith$(\forall i)$ single instance for the n_i number of components in the mobile application, and r_1 be an objective rate of the average overhead of runtime application profiling. Then a_i can be defined as, $a_i = n_i \times r_1$ for $n_i \geq 1$ and $0 < r_1 \leq 1$. Now the total overhead of profiler activation at runtime can be obtained as,

$$a = \sum_{i=1}^{n} a_i, \quad \text{where} \quad a_i > 0, \forall i. \tag{11}$$

4.3 Modeling Application Partitioning Overhead

The overhead of runtime application partitioning depends on the computational intensity of the components being partitioned at runtime. Typically optimizer uses an additional arbitrary rate of CPU as per the intensity of the component for each instance of optimizer activation. Also the overhead of each instance of the optimizer activation vary with the memory size or processing length of the intensive component of the mobile application. Thus, the application partitioning overhead runtime for a single instance can be modeled on the basis of the number of components in the mobile application with a runtime application partitioning rate.

Let b_i be the respective processing overhead of the ith single instance, and r_2 be an objective rate of processing overhead of optimizer activation down to the computational intensity of the operation. In this case b_i can be expressed as $b_i = \max\{y_i, z_i\} \times r_2$ for $y_i > 0, z_i > 0(\forall i)$ and $0 < r_2 \leq 1$. Hence, the total overhead of optimizer activation at runtime can be determined as,

$$b = \sum_{i=1}^{n} b_i \quad \text{for} \quad 0 < a_i \leq 1. \tag{12}$$

Moreover, the total overhead of runtime optimization is the sum of the total profiler overhead a and total optimizer overhead b for resolving the problems of n number of intensive components of the mobile application. Now if t represents the total overhead of runtime partitioning for the mobile application, then the total overhead of runtime optimization for X can be estimated as,

$$t = a + b \tag{13}$$

5 Evaluation and Validation

The intensive mobile application is represented in terms of computational requirements of the application. The computational requirements of the application are represented in terms of memory size and processing length of the components of the application. We investigate

computing resources requirement for the same mobile application in two different scenarios A and B. Scenario A describes computational offloading by employing the contemporary offloading techniques [24, 39] wherein runtime partitioning is deployed for outsourcing computational load to the remote server node. Scenario B explains leveraging application processing services of computational clouds while employing the proposed DEAP model.

Scenario A represents the partitioning of the elastic application X at runtime in which the application is partitioned at runtime for coping the critical condition of excessive computational requirements of the application. However, scenario B represents distributed elastic application wherein the mobile application is organized on the basis of DEAP model. In scenario B the application is organized between explicitly defined DEAP client X_c and DEAP server X_s application. In order to achieve the objective of richness of services and enhance offline usability, DEAP client is enriched with application processing logic of slight intensive components of the application whereas DEAP server is configured with the highly intensive components of the application. Therefore, DEAP client is a distinct framework from traditional thin client applications.

Let τ_p denotes the CPU speed and τ_m denotes the RAM capacity of SMD. The application is tested for Android devices with (ARMv7 CPU having 600 MHz Processor, (470.22 BogoMIPS speed), and 512 MB RAM capacity. Therefore, $\tau_p = 470.22$ MIPS and $\tau_{m=}$ 512 MB. The computational intensity of mobile application is determined by using prototype application for Android devices. The mobile application is composed of five intensive service components. The intensive components of mobile application are as follows:

1. A sorting service component which implements the logic of bubble sorting for linear list of 20,000 integer type values. The computational requirement of the sorting service includes 7.8 KB memory and 402.7 MI processing length.
2. A matrix multiplication service component which implements the logic of computing the product of 300 × 300 size two dimensional array of integer data type values. The computational requirements of the matrix multiplication service include 8.4 KB RAM and 286.4 MI processing length.
3. The power computing service which implements the logic of power computing b^e (is it power) where $b = 2$ and $e = 10^8$. The computational requirements of the power compute service requires 7.8 KB RAM and 279.9 MI processing length.
4. The factorial compute service computes factorial of n, whereas $n = 40$. The computational requirement of the factorial compute service includes 7.9 KB RAM and 51.7 MI processing length.
5. The searching service component of the mobile application searches a linear list of length n, whereas $n = 10{,}000$. The computational requirements of the searching service component include 8.2 KB RAM and 61.135 MI CPU.

By using Eq. 1 the total memory size (T_m) of the mobile application is 40.1 KB. Similarly, by using Eq. 2 the total processing length (T_p) of the mobile application is 1081.9 MI. As, $T_p > \tau_p$, therefore component offloading is required to reduce the processing load on SMD. Profiler and Optimizer are required to be activated three times for separating the larger intensive components (x_3–x_5) from the mobile application. By employing profiling on the Android platform it is found that profiler overhead (a) is 37% and Optimizer overhead is 5% of the computational intensity of the component being partitioned at runtime. By using Eq. 11 the total Profiler overhead is computed as $a = 358.5$ MI and by using Eq. 12 total Optimizer overhead is computed as $b = 48.45$ MI.

Equation 13 indicates that the runtime partitioning overhead is the sum of total profiling overhead and total optimizing overhead. Hence, by using Eq. 13 total runtime partitioning overhead t is =407 MI. Runtime partitioning of the application reduces application processing load of the application to 112.9 MI and memory allocation is reduced to 16.1 KB. However, runtime partitioning requires additional 407 MI for application partitioning which shows additional resources utilization in the computational load distribution process.

By employing DEAP model the intensive mobile application is classified into DEAP client application Xc and DEAP server application Xs. Let the client application is composed of 3 components (x1–x3) and the server application is composed of 2 resources intensive components (x4–x5) which are separated at design time. Let set B is the finite set of components in DEAP client application and set C is the finite set of components in DEAP server application. I am not sure about the following notations, please revise it. We know that $X_c \cup X_s = X \in U$ and $X_c \cap X_s = \emptyset$,. By using Eq. 3 the total memory size of the DEAP client application is computed as $T_m^c = 23.9$ KB and by Eq. 4 the total processing length of the DEAP client application is computed as $T_p^c = 392.8$ MI.

Let a single component of the mobile application is offloaded at runtime; therefore the profiler mechanism is activated for partitioning the intensive component of the mobile application. In this scenario, the overhead of runtime profiling is 103.6 MI and the overhead of Optimization is 13.9 MI by partitioning a single component of mobile application. By using Eq. 13 the total overhead (φ) of runtime partitioning is 117.6 MI. As X_s is the finite set of components configured in the DEAP server application; therefore by using Eq. 5 the memory size (T_m^s) of the DEAP server application is computed as 16.2 KB and by using Eq. 6 the processing length (T_p^s) of the DEAP server application is computed as 689.1 MI.

Asis indicates that in scenario A and scenario B partitioning of the application is employed for reducing computation load on SMD. By using Eq. 7 the total memory size of the elastic application is equal to the sum of the memory size of DEAP client and DEAP server application. As, the total memory size of the elastic application is $T_m = 40.1$ KB, whereas the total memory size of DEAP client application is $T_m^c = 23.9$ KB and the total memory size of DEAP server is $T_m^s = 16.2$ KB. Since, $T_m = T_m^c + T_m^s$ which validates Eqs. 7, 8, 9 and 10.

Similarly, by using Eq. 9 the total processing length of elastic application is equal to the sum of processing length of DEAP client and DEAP server application. We know that the total processing length of the elastic application is $T_p = 1081.9$ MI whereas, the total processing length of DEAP client application is $T_p^c = 392.8$ MI and the total processing length of DEAP server application is $T_p^s = 689.1$ MI. DEAP based deployment of the mobile application reduces: Memory allocation on SMD 40.4% and CPU load 63.7%. In both the scenarios (A&B) the computational load offloaded to cloud server node is identical. However, additional resources utilization in computational offloading reduces 74.6% in scenario B for the reason of eliminating the overhead of runtime partitioning of the intensive component of the mobile application, which indicates the lightweight nature of DEAP framework.

Figure 5 shows the total CPU allocation in scenario A, wherein computational offloading is employed by using the existing techniques [24, 39] and scenario B, wherein DEAP model is employed for computational offloading. The analysis indicates that in scenario A the processing load of the application processing on SMD is 1488.9 MI whereas, in scenario B the processing load of the DEAP client application is 496.4 MI. However, in both scenarios two components of the application are executed on SMD,

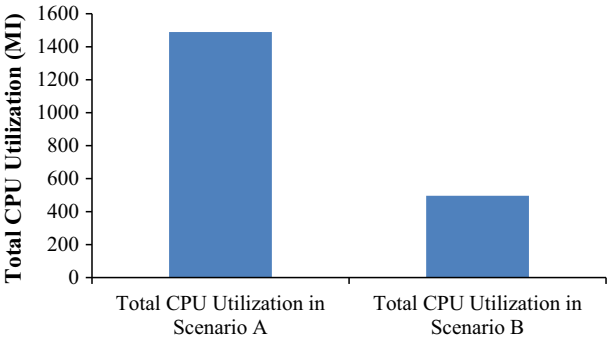

Fig. 5 Comparison of total CPU utilization on SMD in different scenarios

whereas three components of the application are outsourced to cloud server nodes. It shows 66.7% reduction in the CPU utilization for processing the identical workload on SMD in scenario B as compared to scenario A. The decrease in CPU utilization in DEAP model based computational offloading is for the reason of minimizing the instances of runtime component offloading which reduces the overhead of additional CPU utilization in runtime application profiling and optimization.

Figure 6 compares the Profiler overhead in scenario A and B. Profiler overhead is found 358.5 MI in scenario A, wherein three intensive components of the application are offloaded at runtime. The profiler overhead is found 103.6 MI in scenario B, wherein a single intensive component of the application is offloaded at runtime. It shows that the profiler overhead is 28.8% higher in scenario A, as compared to B for the reason of larger number of components in the profiling of elastic application in scenario A.

Similarly, Fig. 7 compares additional overhead of CPU utilization in application partitioning in scenario A and B. Additional CPU utilization in application partitioning is observed 48.5 MI in scenario A; whereas, CPU utilization is found 14 MI in scenario B. It shows that runtime application partitioning involves 28.8% additional CPU utilization. Optimizer overhead is larger in scenario A as compared to scenario B for the reason of partitioning larger number of components with higher computational intensity in scenario A.

Moreover, in [23] we investigated the overhead of application profiling and partitioning. It is concluded that offloading the intensive components of the application at runtime places additional load on mobile device which increases the utilization of computing resources on SMD. As a result, the execution time of the locally executing components of the mobile application is affected adversely. It is found that CPU utilization increases 8% and the execution time of the locally executing services increases 32.6%. It is also found that during the component offloading process at runtime CPU utilization increases 9.8% which shows the additional overhead of application partitioning and component offloading at runtime.

We tested a prototype application on Android platform in the real MCC environment for evaluating the size data transmission and significance of DEAP framework. The prototype application is composed of sort service, matrix multiplication service and power compute service components. Each component of the application is evaluated with 30 different computational intensities. Figure 8 shows the comparison of the size of data transmission in employing the latest existing techniques [24, 39] and DEAP framework for

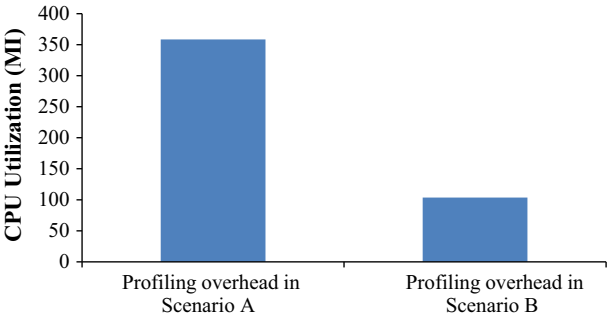

Fig. 6 Comparison of runtime profiling overhead

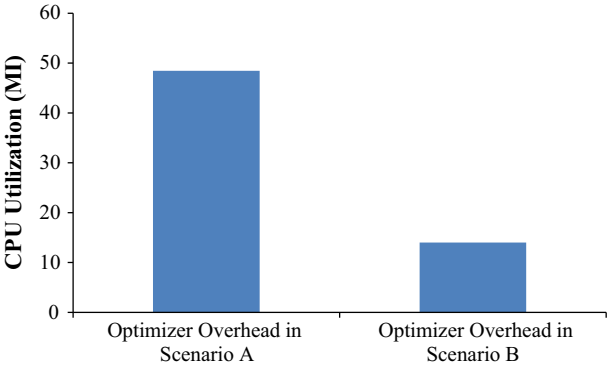

Fig. 7 Comparison of runtime optimization overhead

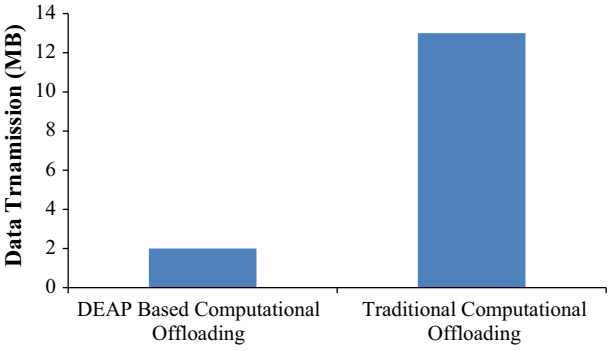

Fig. 8 Comparison of the size of data transmission in traditional and DEAP framework based computational offloading

computational offloading. Data transmission in runtime component offloading for the applications on Android platform involves application package file (.apk) and preferences file (.xml). Application package is the complied binary file of the application whereas, preferences file contains data saved while interrupting the running instance of the

application. Preferences files are transmitted between mobile device and remote server node for exchanging data. It is found that by employing runtime component offloading 13 MB data is transmitted in offloading all the three components of the application. However while employing DEAP framework for computational offloading, 2 MB data is transmitted over the network. It shows that by reducing the instances of runtime components offloading the size of data transmission is reduced up to 84% in DEAP framework.

For evaluating the significance of DEAP framework, the prototype mobile application is executed on local mobile device, traditional offloading techniques [24, 39] and DEAP framework based computational offloading. Figure 9 compares allocation memory on mobile device in local and remote execution of the application. It is found that 31.8 MB RAM is allocated for the processing of application on local mobile device, 45 MB RAM is allocated in employing runtime component offloading, and 9.1 MB RAM is allocated in employing DEAP framework for computational offloading. It shows that by employing DEAP framework for computational offloading memory utilization is reduced by 79.8 and 71.5% as compared to existing techniques [24, 39] and the execution of application on local mobile device respectively.

Figure 10 shows the comparison of CPU utilization on mobile device for different components of the prototype application in local and DEAP based remote execution. It shows that by employing computational offloading the application processing load on the

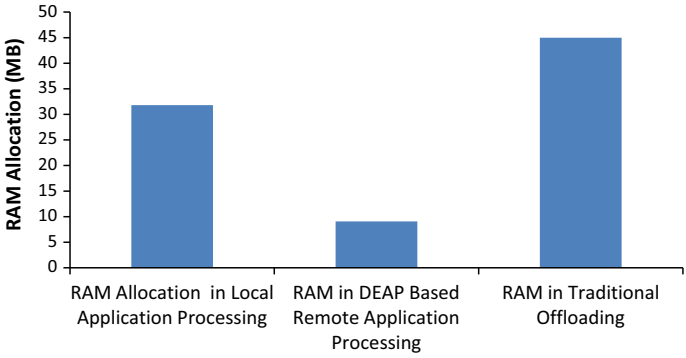

Fig. 9 Comparison of memory allocation on mobile device in local and remote execution of the application

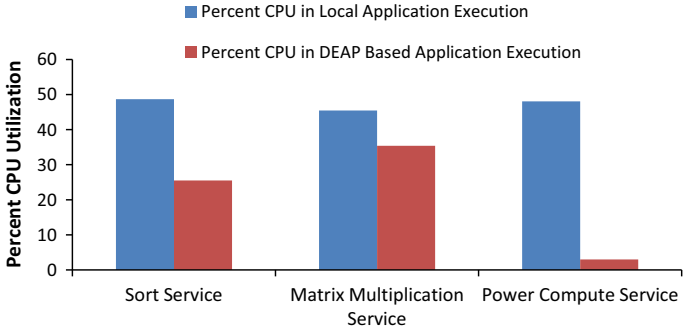

Fig. 10 Comparison of CPU utilization on mobile device in local and DEAP based remote execution

mobile device is decreased, which results in reduction of CPU utilization on mobile device. In executing the prototype application on mobile device, CPU utilization for different components of the application is found 48.7% for sort service, 45.5% for matrix multiplication service and 48% for power computing service. However, by deploying DEAP framework for computational offloading CPU utilization for different components is found as 25.5% for sort service, 35.4% for matrix multiplication service and 3% for power computer service. It shows that in deploying DEAP for computational offloading CPU utilization on mobile device decreases 47% for sort service, 22% for matrix multiplication service and 93.8% for power compute service. Similarly, the duration of CPU utilization on mobile device is decreased 55.8% for sort service, 9.95% for matrix multiplication service and 94% for power computing service. Deploying DEAP framework based computational offloading decreases average CPU utilization mobile by 55.5% and the duration of CPU utilization decreases by 54% which signifies the usefulness of computational offloading for MCC.

There are following three fold advantages of the configuration of preconfigured server components:

1. The separation of intensive components of the mobile application (which are feasible for cloud based execution) at design time minimizes the occurrences of runtime application partitioning. Therefore, the overhead of additional CPU utilization in runtime component partitioning is reduced.
2. The configuration of intensive services/component in cloud server nodes results in the reduction of the data transmission overhead. Since, the intensive components are preconfigured; therefore, client mobile application requires activating remote services by using IPC mechanism instead of runtime component migration. As result, the overhead of component migration is reduced.
3. The services of cloud server can be utilized anytime and anywhere for mobile client applications on demand basis. Therefore, the availability of distributed services is increased.

6 Conclusion

We have proposed a distributed and elastic mobile application as a lightweight alternative for elastic mobile applications. DEAP framework mitigates the deficiencies of current offloading frameworks which are elastic in nature and lack in distributed design perspective which makes the development and deployment of distributed platform resources intensive on SMD. The incorporation of distributed application architecture facilitates in the simple application developmental procedures and lightweight mechanism for accessing the preconfigured services in cloud datacenters on demand basis. The dual operating nature of DEAP framework contributes to the versatility and robustness of the distributed and elastic model for intensive mobile application for MCC. DEAP implements the access of preconfigured services in cloud datacenters as the primary operation procedure, whereas runtime component offloading is implemented as the secondary operating procedure for coping with the dynamic processing load on smart mobile devices. The elasticity attributes of DEAP client enables resources constraint SMDs to cope with the challenges of dynamic application processing loads. Further, the elastic nature of DEAP client contributes to the objectives of offline usability, smart client and rich internet applications for MCC.

Analysis of the DEAP framework shows that separation of intensive components of the application at design time is significant approach for reducing additional resources utilization in computational offloading for MCC. We conclude that DEAP framework is a lightweight and optimal model for leveraging application processing services of computational clouds in MCC.

Acknowledgements This work 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. Shiraz, M., Whaiduzzaman, M., & Gani, A. (2013). A study on anatomy of smartphone. *Computer Communication & Collaboration, 1*(1), 24–31.
2. Abolfazli, S., Sanaei, Z., Gani, A., Xia, F., & Yang, T. L. (2013). Rich mobile applications: Genesis, taxonomy, and open issues. *Journal of Network and Computer Applications.* doi:10.1016/j.jnca.2013.09.009. **(in press)**.
3. Balan, K. R., Gergle, D., Satyanarayanan, M., Herbsleb, J. (2007). Simplifying cyber foraging for mobile devices. In *Proceedings of 5th USENIX International Conference on Mobile Systems, Applications and Services (MobiSys), San Juan, Puerto Rico* (pp. 272–285)
4. Flinn, J., Park, S., & Satyanarayanan, M. (2002). Balancing performance energy, and quality in pervasive computing. In *22nd International Conference on Distributed Computing Systems (ICDCS02), Vienna, Austria* (pp. 217–226).
5. Kristensen, D.M. (2007). Enabling cyber foraging for mobile devices. In *5th MiNEMA Workshop, Magdeburg, Germany* (pp. 32–36)
6. Su, Y.Y., & Flinn, J. (2005). Slingshot: Deploying stateful services in wireless hotspots. In *3rd International Conference on Mobile Systems, Applications, and Services, New York, NY* (pp. 79–92)
7. Kristensen, D. M., & Bouvin, O.N. (2008). Developing cyber foraging applications for portable devices. In *2nd IEEE International Interdisciplinary Conference on Portable Information Devices. Garmisch-Partenkirchen, Germany* (pp. 1–6).
8. Porras, J., Riva, O., & Kristensen, D. M. (2009). *Dynamic resource management and cyber foraging* (Vol. ch. 16, pp. 349–368). Berlin and Heidelberg: Springer.
9. Chun, B., & Maniatis, P. (2009). Augmented smartphone applications through clone cloud execution. In *12th Workshop on Hot Topics in Operating Systems (HotOS), Monte Verita*
10. Satyanarayanan, M., Bahl, P., Cceres, R., & Davies, N. (2009). The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing, 8*(4), 14–23.
11. Mohsen, S., Somayeh, K., & Omid, K. (2011). A survey and taxonomy of cyber foraging of mobile devices. *Communications Surveys & Tutorials, IEEE Communications Society, 14*(4), 1232–1243.
12. Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A., & Buyya, R. (2013). Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open issues. *IEEE Communications Surveys and Tutorials*. doi:10.1109/SURV.2013.070813.00285. **(in Press)**.
13. Whaiduzzaman, M., Sookhak, M., Gani, A., & Buyya, R. (2013). A survey on vehicular cloud computing. *Journal of Network and Computer Applications.*. doi:10.1016/j.jnca.2013.08.004. **(in press)**.
14. Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems, 25*(6), 599–616.
15. Amazon S3. http://status.aws.amazon.com/s3-20080720.html. Accessed on 20th July 2011
16. Google Docs. http://docs.google.com. Accessed on 15th July 2011
17. MobileMe. http://en.wikipedia.org/wiki/MobileMe. Accessed on 15th June 2011.
18. Dropbox. http://www.dropbox.com. Accessed on 15th July 2011.
19. Shiraz, M., & Gani, A. (2012). Mobile cloud computing: Critical analysis of application deployment in virtual machines. In *Proceedings of ICICN 2012, Singapore*.
20. Shiraz, M., Gani, A., & Khokar, H.R. (2012). Towards lightweight distributed applications in mobile cloud computing. In *Proceedings of 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE 2012), China.*

21. Shiraz, M., Gani, A., Khokar, H. R., & Buyya, R. (2012). A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *IEEE Communications Surveys & Tutorials, 15*(3), 1294–1313.
22. Sanaei, Z., Abolfazli, S., Gani, A., & Buyya, R. (2013). Heterogeneity in mobile cloud computing: Taxonomy and open challenges. *IEEE Communications Surveys and Tutorials*. doi:10.1109/SURV.2013.050113.00090. **(in Press)**.
23. Shiraz, M., Ahmed, E., Gani, A., & Han, Q. (2013). Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing. *Journal of Supercomputing*. doi:10.1007/s11227-013-0988-6. **(in Press)**.
24. Hung, H. S., Shih, S. C., Shieh, P. J., Lee, P. C., & Huang, H. Y. (2012). Executing mobile applications on the cloud: Framework and issues. *Computers & Mathematics with Applications, 63*(2), 573–587.
25. Cuervo, E., Balasubramanian, A., Cho, K.D., Wolman, A., Saroiu, S., Chandra, R., & Bahlx, P. (2010). MAUI: Making smartphones last longer with code offload. In *Proceedings of MobiSys'10, San Francisco, California*
26. Zhang, X., Kunjithapatham, A., Jeong, S., & Gibbs, S. (2011). Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. *Mobile Networks & Applications, 16*(3), 270–285.
27. Canepa, H.G., & Lee, D. (2010). A virtual cloud computing provider for mobile devices. In *ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond MCS'10, San Francisco, California*, ACM Press
28. Goyal, S., & Carter, J. (2004). A lightweight secure cyber foraging infrastructure for resource-constrained devices. In *WMCSA 2004 Sixth IEEE Workshop, IEEE Publisher*.
29. Dou, A., & Kalogeraki, V. (2010). MISCO: A MaPreduce framework for mobile systems. In *PETRA'10 Greece,* ACM Press
30. Chun, B. G., & Maniatis, P. (2009). *Augmented smartphone applications through clone cloud execution*. Berkeley: Intel Research.
31. Satyanarayanan, M., Bahl, P., & Caceres, R. (2009). The case for VM-based cloudlets in mobile computing. *IEEE Computing Society, 8*(4), 14–23.
32. Zao, B., Xu, Z., Chi, C., Zhu, S., & Cao, G. (2011). Mirroring smartphones for good: A feasiblity study. *ZTE Communications, 9*(1), 13–18.
33. Chun, G.B., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011). CloneCloud: Elastic execution between mobile device and cloud. In *Proceedings of EuroSys'11 Salzburg Austria ACM Press*
34. Messer, A., Greenberg, I., Bernadat, P., Milojicic, D., Chen, D., Giuli, J. T., et al. (2002). *Towards a distributed platform for resource-constrained devices*. Palo Alto: Hewlett-Packard Company.
35. Giurgiu, I., Riva, O., Juric, D., Krivulev, I., & Alonso, G. (2009). Calling the cloud: Enabling mobile phones as interfaces to cloud applications. In *Proceedings of Middleware'09 the ACM/IFIP/USENIX 10th International Conference on Middleware Urbana Champaign, Illinois*
36. Shiraz, M., Gani, A., Khokhar, H.R., & Ahmed, E. (2012). An extendable simulation framework for modeling application processing potentials of smart mobile devices for mobile cloud computing. In *Proceedings of Frontiers of Information Technology 2012*
37. Application Fundamentals http://developer.android.com/guide/components/fundamentals.html Accessed on 1st November 2013.
38. Calheiros, N. R., Ranjan, R., Beloglazov, A., Rose, D. F. A. C., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software Practice and Experience, 41*(1), 23–50.
39. Shiraz, M., & Gani, A. (2013). A lightweight active service migration framework for computational offloading in mobile cloud computing. *Journal of Supercomputing*. doi:10.1007/s11227-013-1076-7. **(in Press)**.

Dr. Muhammad Shiraz is an Assistant Professor at Department of Computer Science, Federal Urdu University of Arts, Science and Technology Islamabad, Pakistan. He has completed his Ph.D. Degree with Distinction from University of Malaya, Malaysia in 2013. He completed Masters in Computer Science from Allama Iqbal Open University Islamabad, Pakistan in 2007 and under graduation from CECOS University of Information Technology and Emerging Sciences Peshawar, Pakistan with the distinction of Gold medal. He is an active researcher in the Mobile Cloud Computing Research Group at Faculty Computer Science and Information Technology University Malay Kuala Lumpur. His areas of interest include distributed applications design for ubiquitous networks, distributed systems, lightweight applications, smart client applications and optimization strategies, mobile cloud computing.

Abdullah Gani is Professor at the Department of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, Malaysia. His academic qualifications were obtained from the University of Hull, UK for bachelor and master degrees, and the University of Sheffield, UK for Ph.D. He has vast teaching experience due to having worked in various educational institutions locally and abroad—schools, teaching college, ministry of education, and universities. His interest in research started in 1983 when he was chosen to attend the Scientific Research Course in RECSAM by the Ministry of Education, Malaysia. More than 100 academic papers have been published in conferences and respectable journals. He actively supervises many students at all level of study—Bachelor, Master and Ph.D. His interest of research includes self-organized system, reinforcement learning, wireless-related networks. He is now working on mobile cloud computing with High Impact Research Grant of USD 500,000 (RM 1.5 M) for the period of 2011–2016. He is a senior member of IEEE. Currently, he is a director of the Centre for Mobile Cloud Computing Research, which focuses on high impact research. He is also a visiting Professor at the King Saud University, Saudi Arabia as well as serves as Adjunct Professor at the COMSATS Institute of Information Technology, Islamabad, Pakistan. He also serves as a visiting professor at the University Malaysia Sabah, Kota Kinabalu, Sabah. Malaysia (2015–2017). He serves as a chairman of Industry Advisory Panel for Research Degree Program at UNITEN, Malaysia (2015–2017).

Rashid Khokhar is working as a Lecturer in School of Computing and Mathematics at Charles Sturt University, Australia. His research interests include mobile cloud computing, sensor networks and vehicular networks.

Azizur Rahman is working as Senior Lecturer in School of Computing and Mathematics at Charles Sturt University, Australia. His research interests include data mining, Big-data analysis, statistical models, mobile networks and cloud computing.

Mohsin Iftikhar got his B.Sc. in Electrical Engineering from University of Engineering and Technology, Lahore Pakistan, Masters of Engineering Science in Telecommunications from UNSW and Ph.D. in Advanced Networks from University of Sydney, Australia in 1999, 2001 and 2008 respectively. He is currently working as senior lecturer in School of Computing and Mathematics at Charles Sturt University, Australia. His research interest include stochastic modelling, queuing theory, mobile computing, polling models, cloud computing and software defined networks.

Naveen Chilamkurti is currently Acting Head of Department, Computer Science and Computer Engineering, La Trobe University, Melbourne, VIC, Australia. He obtained his Ph.D. degree from La Trobe University. He is also the Inaugural Editor-in-Chief for International Journal of Wireless Networks and Broadband Technologies launched in July 2011. He has published about 165 Journal and conference papers. His current research areas include intelligent transport systems (ITS), wireless multimedia, wireless sensor networks, and so on. He currently serves on the editorial boards of several international journals. He is a Senior Member of IEEE. He is also an Associate editor for Wiley IJCS, SCN, Inderscience JETWI, and IJIPT.