Journal of Network and Computer Applications 52 (2015) 52–68

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges

Ejaz Ahmed [a,*], Abdullah Gani [a], Mehdi Sookhak [a], Siti Hafizah Ab Hamid [a], Feng Xia [b]

[a] Mobile Cloud Computing Research Lab, Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia
[b] Mobile and Social Computing Laboratory, School of Software, Dalian University of Technology (DUT), China

ARTICLE INFO

Article history:
Received 27 June 2014
Received in revised form
30 November 2014
Accepted 28 February 2015
Available online 10 March 2015

Keywords:
Mobile cloud computing
Cloud-based mobile applications
Application optimization
Cloud-based mobile application execution frameworks

ABSTRACT

In Mobile Cloud Computing (MCC), migrating an application processing to the cloud data centers enables the execution of resource-intensive applications on the mobile devices. However, the resource-intensive migration approaches and the intrinsic limitations of the wireless medium impede the applications from attaining optimal performance in the cloud. Hence, executing the application with low cost, minimal overhead, and non-obtrusive migration is a challenging research area. This paper presents the state-of-the-art mobile application execution frameworks and provides the readers a discussion on the optimization strategies that facilitate attaining the effective design, efficient deployment, and application migration with optimal performance in MCC. We highlight the significance of optimizing the application performance by providing real-life scenarios requiring the effective design, efficient deployment, and optimal application execution in MCC. The paper also presents cloud-based mobile application-related taxonomies. Moreover, we compare the application execution frameworks on the basis of significant optimization parameters that affect performance of the applications and mobile devices in MCC. We also discuss the future research directions for optimizing the application in MCC. Finally, we conclude the paper by highlighting the key contributions and possible research directions in cloud-based mobile application optimization.

© 2015 Elsevier Ltd. All rights reserved.

Contents

* Corresponding author.
E-mail addresses: imejaz@siswa.um.edu.my (E. Ahmed), abdullahgani@ieee.org (A. Gani), m.sookhak@ieee.org (M. Sookhak), siti.hamid@ieee.org (S.H.A. Hamid), f.xia@ieee.org (F. Xia).

http://dx.doi.org/10.1016/j.jnca.2015.02.003
1084-8045/© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

During the last decade, the advancements of network technologies and the growth of computational devices have turned the dream of ubiquitous computing into reality. The favourable developments have provided a driving force for a number of emerging e-applications such as e-learning, e-commerce, e-tourism guides, e-health, and internet gaming. Recently, with the advent of wireless technologies (Chin et al., 2014; Ahmed et al., 2015b,c) and mobile devices, e-application paradigm is shifted towards m-application paradigm. Hence, a range of m-applications, such as m-learning (Motiwalla, 2007), m-health (Cruz and Barros, 2005), m-guides (Oppermann and Specht, 1999), m-gaming (Ballagas et al., 2007), and mobile worker applications (Mazzoleni and Tai, 2007), are now part of mobile user's application suit. The mobile users expect to run m-applications with identical performance level as they get for the similar applications running on the stationary computer systems. However, the mobile devices are resource-constrained devices that cannot provide the same level of user experience.

In this context, the computation migration is endeavoured as a significant software-level solution that mitigates resource constraints of mobile devices by migrating applications to available stationary computers (Huerta-Canepa and Lee, 2010; Marinelli, 2009; Goyal and Carter, 2004; Cuervo et al., 2010; Kovachev et al., 2012b; Verbelen et al., 2012b; Fesehaye et al., 2012; Yang et al., 2008; Chun and Maniatis, 2009). The computational migration is handled by the application execution frameworks. Similar to other research areas such as communication networks (Shamshirband et al., 2015; Zheng, 2008) and distributed systems such as cloud computing (Ning et al., 2013), the optimization techniques are also widely used in application execution frameworks of MCC. The execution frameworks consider diverse optimization objective functions as follows: saving processing power, efficient bandwidth utilization, and minimizing energy consumption. In short, the frameworks are designed to optimize the execution cost. The overall aim of all such approaches is to enable the compute-intensive mobile applications on resource-constrained mobile devices. The execution of compute-intensive components of a mobile application in mobile cloud computing (MCC) involves the complex application partitioning at different granularity levels and component migration to the cloud server node (Ahmed et al., 2015a). Such delay-inducing and resource-intensive mechanisms adversely affect the user experience. Therefore, it is imperative to employ lightweight procedures for optimal execution of intensive mobile applications in MCC. The optimal execution refers to the state of the application execution in MCC that can deliver enhanced performance as compared to local execution with minimum cost and low overhead. The effective design of an application ensures that the finished design incurs low cost, high reliability, and the excellent performance. The efficient deployment ensures that non-collocated components of mobile application in MCC have minimal dependency on each other; thereby, reducing the operational overhead and execution cost.

Although several survey papers (Abolfazli et al., 2013; Fernando et al., 2013; Dinh et al., 2013; Kumar et al., 2013; Khan et al., 2014; Rahimi et al., 2014) have studied different aspects of leveraging the cloud services to augment the capabilities of mobile devices, application execution optimization in MCC is still not investigated. In Abolfazli et al. (2013), we have comprehensively studied cloud-based mobile augmentation and discussed various methods to augment the potential of mobile devices. In Kumar et al. (2013) and Khan et al. (2014), the authors have reviewed state-of-the-art distributed application offloading frameworks for smart mobile devices. The research works in Fernando et al. (2013), Dinh et al. (2013), and Rahimi et al. (2014) present the comprehensive surveys on MCC that cover the application, architectures, open issues, and challenges. However, this paper is the first research effort that surveys the state-of-the-art application execution frameworks to identify optimization approaches employed by the application designers, classify the optimization approaches, and highlight the challenges involved in attaining the application optimization.

The contribution of the paper includes (a) survey of the state-of-the-art application execution frameworks in MCC; (b) identification of optimization approaches related to application design, deployment, and execution in MCC; (c) classification and presentation of identified approaches in the form of taxonomies; (d) comparison of the state-of-the-art application execution frameworks; and (e) identification of open research challenges in optimizing the application design, deployment, and execution of application in MCC. The comparison highlights the commonalities and differences among the state-of-the-art application execution frameworks on the basis of significant parameters that affect the performance of application. Some of the parameters are transmission delay, Quality of Service (QoS) support, profiler overhead, scalability, and operational cost. We also present different application migration metrics that are usually optimized in migration decision. Finally, we discuss the future research required to optimize the application design, deployment, and execution in MCC.

The paper is organized into the following sections. Section 2 introduces the fundamental concepts of MCC, cloud-based mobile application and its execution in MCC, and application execution optimization. Section 3 discusses the real-life scenarios that highlight the requirements of application optimization in MCC. Section 4 presents the literature survey for current application execution

frameworks for MCC and provides the comparative study of current application execution frameworks to highlight the merits and demerits of existing frameworks. The application-centric taxonomies are presented in Section 5 that covers (a) application optimization strategies, (b) application execution operations, and (c) application execution frameworks. In Section 6, suitability of various metrics is investigated to assist the frameworks designers in selection of suitable metric for attaining the optimized application performance. Future research directions on the basis of literature survey are discussed in Section 7. Finally, open research challenges are presented in Section 8 and Section 9 draws the conclusions by presenting the summary and insights for the readers.

2. Background

This section provides a brief background of MCC, cloud-based mobile application and its execution in MCC. Moreover, the section also briefly discusses the application execution optimization to provide the fundamental knowledge to reader.

2.1. Mobile cloud computing

MCC is the latest distributed computing model, which extends the widespread services and resources of computational clouds for mitigating resource limitations in mobile devices. MCC reduces the development and execution cost of mobile applications and enables mobile user to acquire new technology conveniently on demand basis. Therefore, the model is attracting the attention of enterprises as a profitable business. MCC focuses on alleviating resource limitations in mobile devices by employing different augmentation strategies; such as screen augmentation, energy augmentation, storage augmentation, and application processing augmentation of mobile device (Abolfazli et al., 2013). MCC employs the storage services of computational clouds to enable off-device storage and accesses the application processing services of cloud server nodes to enable compute-intensive applications on mobile devices. However, an optimized access to the contents which are stored in cloud data centers is still challenging research perspectives.

2.2. Cloud-based mobile application and its execution in MCC

Cloud-based mobile applications can run on mobile devices as well as on the cloud. The cloud-based mobile applications consist of two types of components: transferable and non-transferable. The transferable components are memory-intensive or compute-intensive and do not interact with the mobile hardware, whereas non-transferable components are designed and implemented for especial functionality, such as hardware access, user-interaction, and security-related tasks (Cuervo et al., 2010). The process of dividing the application components into two sets, transferable and non-transferable, is called partitioning. Usually, the partitioning is performed in three different ways; statically, dynamically, and semi-dynamically.

The mobile application execution in MCC can be illustrated by a state diagram as shown in Fig. 1. The execution of the application starts when a user clicks the application icon. The application enters into the running state where it performs different tasks. To migrate the application into the cloud, the execution framework pauses the running application. The application saves its running states when the execution framework triggers pause. Then the execution control transfers into the paused state. The application and running states are migrated to the cloud server where the application is resumed and reconfigured using the saved states. The execution control enters into the running state. On completion of the execution on the cloud server, the results are pushed back to the mobile device, where the application resumes its execution on the mobile device. Finally, on the completion of the execution, application stops and enters into the terminated state.

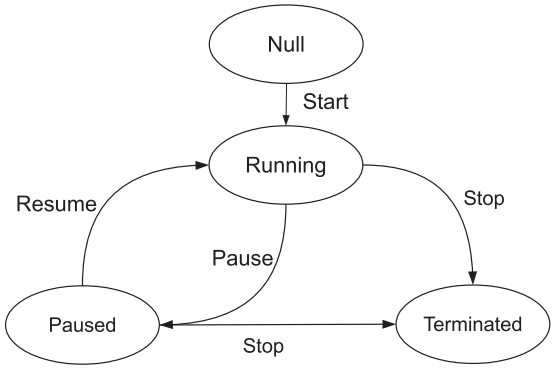

Fig. 1. Application execution state diagram (Ahmed et al., 2015a).

2.3. Application execution optimization

The application execution optimization is vital for MCC because of resource-constrained nature of mobile devices, distributed execution environment, and intrinsic limitations of communication medium. The performance of an application execution in MCC can be improved by attaining the effective design of the application, by efficient deployment, and by optimal application execution in the distributed environment. Hence, the application execution in MCC requires effective design, efficient deployment, and optimal execution in MCC. The effective design of an application ensures that the application incurs low cost, high reliability, and the excellent performance. The efficient deployment ensures that non-collocated components of mobile application in MCC have minimal dependency on each other; thereby, reducing the operational overhead and execution cost. The optimal execution enables uninterrupted and continuous application execution with minimal user involvement. The application execution frameworks are also designed with the aim to optimize different performance metrics of an application. Example includes the following: Mobile Assistance Using Infrastructure (MAUI) (Cuervo et al., 2010) is designed to optimize the energy consumption of the mobile device.

3. Motivation

This section presents the application scenarios in the domain of MCC for providing the motivation to do research in the area. The application execution optimization is demanded in various scenarios, such as image processing, voice recognition and translation, m-gaming, cooperative spectrum sensing in cognitive radio cloud networks, because of resource-intensive nature of the application and intrinsic limitations of the mobile devices and communication medium.

3.1. Image processing

The scenario discussed in Huerta-Canepa and Lee (2010) is an example of compute-intensive time-critical image processing mobile application. In the example scenario, “Peter who is a foreign visitor in South Korea finds an interesting exhibit, with a description written in Korean language, during his visit to a museum. He does not understand the description written in Korean. He takes a snapshot of the exhibit and wishes to process whole text image using a character recognition program. The execution of program involves high computation so resource-constrained mobile device cannot run it. He can execute the software in three different ways: in (a) remote cloud,

(b) cloudlet, and (c) virtual cloud or mobile ad-hoc cloud. For simplicity, we consider that he opts mobile ad-hoc cloud to run the application. He discovers mobile devices of other visitors who have same objective to translate the description. He augments his mobile device resources by utilizing idle mobile device resources of other travellers in the vicinity. He forms a mobile ad-hoc cloud network. Then, the mobile ad-hoc cloud runs the application for text extraction and language translation".

The mobile devices have limited resources and devices are battery powered. Therefore, the application execution process should efficiently consume the available resources and conserves the battery power. Moreover, Peter and the other travellers are in the museum for a short period and they have to extract number of descriptions, so, it is vital to minimize the application execution time for such real-life problems. The disruption because of application migration should also be minimized to provide the acceptable user experience. The optimal application execution frameworks in this scenario can empower them to explore a number of exhibits during their short stay with less overhead involvement of mobile device. The application migration process should also aim to minimize the consumption of energy during the migration. However, there are a number of challenges involved in realizing the optimal application execution in this scenario. The device discovery to form mobile ad-hoc cloud is a main challenge to realize the optimized application execution in this scenario. The application partitioning, task scheduling, application migration, authentication, authorization, and ensuring privacy of personal data on mobile device are other common challenges in realizing the optimal application execution.

3.2. Voice recognition and translation

The importance of realizing the application execution optimization in MCC can also be further highlighted by a scenario where Peter, a foreign visitor, finds some local Koreans in the museum and wants to have a discussion with the locals about the exhibits. He does not know Korean language and the locals do not know the English language which Peter can understand. Now Peter wants to use the voice recognition and translation software to enable a communication with the locals. The software is heavyweight to run on the mobile device so he has to rely on the cloud. He can run the software in three different ways in MCC environment: (a) mobile ad-hoc cloud, (b) cloudlet, and (c) remote cloud. In this case, we consider that he selects the remote cloud to run the software. He has to perform number of tasks before the actual execution of the software such as cloud server discovery, services scanning, authentication, and other tasks for establishing the application execution platform in MCC. All these tasks need to be optimized to improve the overall application execution time and overhead. As Peter and the locals are in the museum for a shorter period they cannot tolerate the delay involved in application execution in the cloud. In order to sustain the usability of voice recognition and translation software, the execution time in the cloud should be minimized. The mobile devices are resource-constrained devices so the application execution in MCC should use minimal device resources.

3.3. M-gaming

Cloud-based m-gaming is another example for realizing the need of application execution optimization in MCC. Peter is playing a game on his mobile device while waiting for meal in a restaurant. Suddenly, battery power of the mobile device decreased and the mobile device is going to be turned off by prompting user with alert message. This situation disrupts execution of the application on the mobile device. Instead, if the cloud-based execution framework is deployed on the mobile device, then Peter can offload the game either to nearby server or to a remote cloud when he wants to play. The offloading of the game to nearby server or to the cloud can extend the battery power life of the mobile device and Peter can play game without any disruption. However, the offloading decision is based on the residual battery power of the mobile device. In this scenario, we consider offloading of game in nearby available server. The discovery of server and services provided by the server and their authentication involve time-intensive and compute-intensive processes, which need to be optimized to use the nearby available resources in a user-transparent way (Ahmed et al., 2013).

3.4. Cooperative spectrum sensing in cognitive radio cloud networks

In Wu et al. (2012), authors have integrated MCC with cognitive radio networks to process huge amount of spectrum sensing data to generate one complete report for a primary user detection in proximity. The time-sensitive nature of scenario demands high computational resources along with fast authentication to detect the masqueraders and a lightweight data integrity mechanism to check data forgery. The incorrect sensing report may result in false alarm and mis-detection of primary user. The time-sensitive nature of spectrum sensing demands optimized execution of cooperative sensing algorithm in MCC environment and resource-constrained nature of mobile devices requires efficient utilization of resources. The communication medium limitations also require minimizing the communication overhead and cost.

4. State-of-the-art cloud-based mobile application execution frameworks

The application execution frameworks are designed to run in diverse environment and fulfill the requirements of various types of applications. The main aim of the application execution frameworks in MCC is to augment the resources of mobile devices by leveraging the resources and services of the cloud. However, the number of parameters involved in optimization varied from framework to framework. Based on the number of optimization parameters involved, we have classified the state-of-the-art cloud-based mobile application execution frameworks into three main categories: mono-objective optimization-based frameworks, bi-objective optimization-based frameworks, and multi-objective optimization-based frameworks.

4.1. Mono-objective optimization-based frameworks

The mono-objective optimization-based frameworks mainly focus to optimize a single objective. The mono-objective optimization is simplest form of optimization but it cannot fit to diverse running environment and cannot fulfill the requirements of various applications. Herein, we are presenting the frameworks that aim to optimize a single objective:

(*a*) *CloneCloud:* In Chun et al. (2011), CloneCloud, a flexible application partitioner, is proposed that empowers mobile applications to seamlessly offload part of it into the remote cloud. The system employs dynamic profiling and static analysis to partition the mobile application. The main goal of partitioning is to optimize overall execution cost. The execution cost comprises computation cost, Comp(E), and migration cost, Migr(E). The energy consumption cost consists of CPU activity, display state, and network state:

$$C(E) = \mathrm{Comp}(E) + \mathrm{Migr}(E) \tag{1}$$

$$\mathrm{Comp}(E) = \sum_{i \in E, m} [(1 - L(m))I(i,m)C_c(i,0) + L(m)I(i,m)C_c(i,1)] \tag{2}$$

The computation cost takes value from the clone cost variables $C_c(i;1)$ when the method runs on the clone of the mobile device, or from the mobile device cost variables $C_c(i;0)$. The migration cost sums the individual migration cost $C_s(i)$ of those invocations whose methods have migration points:

$$\text{Migr}(E) = \sum_{i \in E,m} R(m) I(i,m) C_s(i) \tag{3}$$

Eq. (4) models the constraint that two methods calling directly each others cannot be on the same location. Eq. (5) shows that method annotated to run on the mobile device run only on the mobile device. Eq. (6) ensures that methods dependent on the same class C's native state are collocated. Eq. (7) ensures that all methods transitively called by a migrated method must not be migrated:

$$L(m_1) \neq L(m_2), \ \forall m_1, m_2 : \text{DC}(m_1, m_2) = 1 \wedge R(m_2) = 1 \tag{4}$$

$$L(m) = 0, \ \forall m \in V_M \tag{5}$$

$$L(m_1) = L(m_2), \ \forall m_1, m_2, C : m_1, m_2 \in V_{Nat_C} \tag{6}$$

$$R(m_2) = 0, \ \forall m_1, m_2 : \text{TC}(m_1, m_2) = 1 \wedge R(m_1) = 1 \tag{7}$$

$R(m)$ is a decision variable that represents the insertion of the migration point in method 'm' at entry and exist of a method, if $R(m)$ is 1 partitioner will insert the migration point otherwise method will remain unmodified. The relationship between two methods is represented by $\text{DC}(m_1, m_2)$ and $\text{TC}(m_1, m_2)$ where $\text{DC}(m_1, m_2)$ read as "method m_1 directly calls method m_2" and $\text{TC}(m_1, m_2)$ read as "method m_1 transitively calls method m_2. $L(m)$ is an auxiliary decision variable that represents the location of a method m. V_M and V_{Nat_C} represent the set of methods and annotated methods of class C, respectively.

(*b*) *Data stream application partitioning framework*: A framework for optimal partitioning of data stream application to maximize the speed/throughput in processing is proposed in Yang et al. (2012). The partitioning problem is modeled as the problem of assigning a set of components of dataflow graph to the resources with an objective to maximize the data stream application throughput. To make the framework a lightweight, the optimization solver is kept at cloud side. The objective function is defined as follows:

$$\max_{x_i, y_{i,j}} \text{TP} = \frac{1}{t_p}, \quad i,j \in \{0, 1, \ldots, v+1\} \quad \text{where}$$

$$t_p = \max\left\{ \max_{i \in V} \left(x_i . \frac{s_i}{\eta p} \sum_{i \in V} x_i \right), \ \max_{(i,j) \in E} \left(\frac{d_{i,j} (x_i - x_j)^2}{y_{i,j}} \right) \right\}$$

$$\text{s.t.} \begin{cases} \sum_{i,j \in E} y_{i,j} (x_i - x_j)^2 = B, \\ y_{i,j} > 0, \\ x_0 = 1, \\ x_{v+1} = 1, \\ x_i = 0 \text{ or } 1, \quad i \in \{1, 2, \ldots, v\} \end{cases} \tag{8}$$

where p and η are the CPU's capability of the mobile device and the percentage of the mobile device ideal CPU resource, respectively. In Eq. (8), B is the bandwidth. The variables x_i is the decision variable that can get value either 1 or 0. The $x_i = 1$ means the component i runs on the mobile device; otherwise $x_i = 0$ runs on the cloud. The variable $y_{i,j}$ represents the wireless bandwidth dedicated to the channel (i,j). The two virtual nodes 0 and $v+1$ are added to satisfy the constraint that data is originated from and delivered to the mobile device. The variables $d_{0,1}$ and $d_{v,v+1}$ represent the size of unit of input data and output data, respectively. However, the genetic-based partitioning algorithm used by the framework does not assure a constant optimization response time.

(*c*) *Computational offloading dynamic middleware* (*CODM*): In Verbelen et al. (2011), a middleware framework for selection of deployment of a configuration with best quality considering current connectivity and available resources is presented. The main unit of deployment is bundle. For each bundle, the framework provides multiple configurations with different levels of resource's demand and offers different quality levels.

The best configuration is modeled by the objective function:

$$\max_{configurations} \sum_i w_i$$

$$\text{s.t.} \begin{cases} \forall m : \sum_i X_{im} \times w_i \leq M_m \\ \forall i : \sum_m X_{im} = 1 \end{cases} \tag{9}$$

The variable w_i represents the weight of ith component that depicts the quality of the component. The decision variable X_{im} value is equal to 1 if it is assigned to machine otherwise it is zero. First constraint ensures that the sum of the weights of all bundles deployed on the device or the server does not exceed the maximum allowed limit. Second constraint ensures that each bundle of the configuration can be deployed on either the mobile device or the server. The framework can dynamically adapt the configuration and deployment, and gracefully degrades the quality in case of connection lost. However, the framework requires different configurations and deployments of application. Moreover, the framework also requires Open Service Gateway initiative (OSGi) support in remote cloud.

(*d*) *Hyrax*: Hyrax (Marinelli, 2009) leverages the available resources from a cluster of mobile devices in a local proximity to run the compute-intensive tasks. Hyrax employs fault tolerance mechanism of Hadoop to reduce the frequent disconnections of mobile servers. Hyrax also provides accessibility to the remote clouds in case the nearby mobile resources are not sufficiently available. Hyrax is developed based on Hadoop for Android mobile devices. The server employs two client side processes of MapReduce, namely NameNode and JobTracker to coordinate the computation process on a group of mobile devices. The mobile device employs two Hadoop processes, namely TaskTracker and DataNode to receive the tasks from the JobTracker. The mobile devices connect with the server and other mobile devices via IEEE 802.11g technology. The Hyrax transparently uses distributed resources and provides interoperability across heterogeneous platform. However, the Hyrax has high overhead because of the complexity of Hadoop algorithm.

(*e*) *Virtualized mobile replicas offloading architecture* (*VMROA*): In Chun and Maniatis (2009), the authors propose an offloading architecture using loosely synchronized virtualized mobile replicas in the cloud. The execution in the offloading architecture involves three phases. The architecture demands less bandwidth due to incremental checkpoint system replication. There are three main components of architecture, namely replicator, controller, and augmenter. The replicator is responsible for synchronizing the changes in phone software and states of the clone. The controller on the mobile device starts an augmented execution and reintegrates the result back to mobile device. The augmenter on the clone side is responsible for local execution and returning of results. The VMROA reduces the synchronization overhead by opportunistically synchronizing the mobile device and cloud server. However, the execution environment migration creates issues of security, privacy, and management in the cloud.

(*f*) *MAUI*: In Cuervo et al. (2010), MAUI, an energy-aware method level offloading mechanism for mobile applications in the cloud is proposed. MAUI employs both static and dynamic partitioning. First of all, the programmer annotates the method as a remotable which does not implement the following

Table 1
Comparison of mono-objective optimization-based frameworks based on objectives.

Mono-objective optimization-based frameworks	Optimizing execution cost	Maximizing throughput	Optimizing deployment	Reducing network latency	Transparent resource augmentation	Minimizing energy consumption
CloneCloud (Chun et al., 2011)	✓	✗	✗	✗	✗	✗
Data stream application partitioning framework (Yang et al., 2012)	✗	✓	✗	✗	✗	✗
CODM (Verbelen et al., 2011)	✗	✗	✓	✗	✗	✗
Hyrax (Marinelli, 2009)	✗	✗	✗	✓	✗	✗
VMROA (Chun and Maniatis, 2009)	✗	✗	✗	✗	✓	✗
MAUI (Cuervo et al., 2010)	✗	✗	✗	✗	✗	✓

Table 2
General comparison of mono-objective optimization-based frameworks.

Application execution frameworks	Network latency	QoS support	Profiler overhead	Scalable	Programmer support	Cloud usage overhead	Energy consumption	Operational cost
CloneCloud (Chun et al., 2011)	Low	n/a	High	No	No	High	Low	High
Data stream application partitioning framework (Yang et al., 2012)	No	No	Med.	Yes	No	Med.	Low	Low
CODM (Verbelen et al., 2011)	Low	Yes	Low	No	Yes	Low	Low	Low
Hyrax (Marinelli, 2009)	Low	No	n/a	Yes	n/a	Low	Low	Med.
VMROA (Chun and Maniatis, 2009)	Low	n/a	Low	Yes	n/a	High	Low	High
MAUI (Cuervo et al., 2010)	Low	No	High	No	Yes	Low	Low	Low

Med. – Medium, n/a – Not Applicable.

functionalities: (a) the user interface implementation and (b) input/output implementation. Thereafter, the proposed system automatically identifies remotable, non-remotable methods, and states of remotable method, then automatically performs migration. MAUI uses timeout mechanism to detect the failures in connection with the server. MAUI profiler performs three types of profiling namely device profiling, program profiling, and network profiling. MAUI formulates the execution problem as a 0–1 integer linear programming (ILP) problem:

$$\text{maximize} \sum_{v \in V} I_v \times E_v^l - \sum_{(u,v) \in E} |I_u - I_v| \times C_{u,v}$$

$$\text{such that}: \sum_{v \in V} ((1 - I_v) \times T_v^l) + (I_v \times T_v^r)) + \sum_{(u,v) \in E} (|I_u - I_v| \times B_{u,v}) \leq L \quad \text{and} \quad I_v \leq r_v, \quad \forall v \in V \tag{10}$$

where I_v is the decision variable: $I_v = 0$ if method is executed locally, otherwise $I_v = 1$. The E_v^l and T_v^l denote the energy and time required to execute the method locally, respectively. The $B_{u,v}$ represents the necessary program state when u calls v. The parameter $C_{u,v}$ represents the energy cost of transferring states, and parameter r_v indicates that the method is marked remotable or not. Although MAUI significantly improves the energy consumption of mobile device and incorporates the user mobility and network dynamics, it does not address scalability and does not provide the QoS features.

Table 1 presents the comparative summary of mono-objective optimization-based frameworks considering the objective. The general comparison of these frameworks is presented in Table 2.

4.2. Bi-objective optimization-based frameworks

The bi-objective optimization-based frameworks mainly focus to optimize the two objectives. The bi-objective optimization-based frameworks have relatively higher complexity than mono-objective optimization-based frameworks. These frameworks incorporate two diverse parameters in the objective function. The bi-objective optimization-based frameworks can better fit to diverse running environment and can fulfill the some requirements of applications in MCC. Herein, we are presenting the frameworks that aim to optimize two objective functions:

(*a*) *Virtual machine* (*VM*)*-based cloudlet*: In Satyanarayanan et al. (2009), the authors highlight the limitations of cloud computing. The WAN latency, jitter, packet losses are some of the limitations, which hurt the usability of time sensitive applications. Using cloudlet in local proximity reduces latency and provides a single hop, high bandwidth wireless access to the resource-rich cloudlet. It provides pre-use modifications and after-use clean-up, which ensures that the infrastructure is restored in faultless states after use. There are two approaches that can deliver VM state to cloudlet. One is VM migration and the other approach is dynamic VM synthesis. In VM migration approach, an executing VM is paused; all its states are transferred to the cloudlet and then VM execution again resumed at the cloudlet. In the dynamic VM synthesis approach, mobile device sends a small VM overlay to the cloudlet that already has the base VM from which overlay VM was derived. The cloudlet infrastructure then derives the launch VM by applying overlay to the base VM. The proposed solution alleviates the migration overhead by employing the dynamic synthesis. Moreover, the performance of dynamic synthesis is further improved by employing parallel processing and by employing caching and pre-fetching techniques. However, privacy and access control are issues with migration of entire application execution environment that needs to be addressed.

(*b*) *AIOLOS*[1]**:** In Verbelen et al. (2012a), AIOLOS, a mobile middle-ware framework is proposed, which has an adaptive offloading decision engine that considers dynamic available resources of server and varying network conditions in its offloading decision. The framework estimates execution time for each method call at both local and remote execution considering argument size. Thereafter, on basis of the result it selects the execution location. The offloading decision algorithm goal is to optimize local execution time and energy consumption. A history-

[1] aiolos is an ancient Greek word meaning “quickly changing, adapting”.

based profile is used for a service method to calculate the local execution time, which incorporates speedup processor α, network bandwidth β, and latency γ:

$$\overline{T}_{remote} = \frac{1}{\alpha} \times \sum_{i \in RM} (\overline{T}_{CPU,local_i}) + \frac{1}{\beta} \times (A+R) + \gamma + \sum_{j \in CM} \left(\overline{T}_{CPU,local_j} + \frac{1}{\beta} \times (A_j + R_j) + \gamma \right) \quad (11)$$

The parameters A and R represent the argument size and estimated return size respectively.

$$\overline{E}_{saved} = E_{CPU} \times \sum_{i \in RM} (\overline{T}_{CPU,local_i}) - E_{TR} \times A - E_{RCV} \times R - \sum_{j \in CM} (E_{RCV} \times A_j + E_{TR} \times R_j) > 0 \quad (12)$$

where E_{CPU}, E_{TR}, and E_{RCV} represent the energy consumed per time unit by the CPU, the energy cost for transmitting and receiving a byte, respectively. AIOLOS can only perform better in the scenarios where previous history of cloud server is available.

(*c*) *Calling the cloud*: A middleware framework is presented in Giurgiu et al. (2009) that dynamically partitions an application and offloads the partitions to the cloud. The partitioning process comprises two steps. Firstly, an application is modelled as a data flow graph that consists of various software modules. Thereafter, a partitioning algorithm computes the optimal cut, which optimizes the given objective function. The optimization objective function minimizes the interaction latency between the mobile device and the cloud server while taking care of the exchanged data overhead:

$$\min O_{C_c} = \min \left(\sum_{i=1}^{t<k} \sum_{j=1}^{w} \frac{\text{in}_{ij} + \text{out}_{ji} * f_{ij}}{\alpha} + \sum_{i=1}^{k} \frac{\text{code\_size}_i}{\beta} + \sum_{i=1}^{w<s} \text{proxy\_cost}_i \right) \quad (13)$$

where the first part of Eq. (13) models the cost of data exchange between non-collocated components. The parameter f_{ij} represents the frequency of data exchange between non-collocated components. The α and β represent the capacities of communication and computational resources, respectively. The second term represents the computation cost on the mobile device. The last term represents the cost of creating the proxies for interaction with w remote bundles. The framework reduces memory consumption, communication cost, and interaction time. However, the framework employs compute-intensive migration process, which is involved in dynamic analysis, profiling, synthesis, runtime partitioning and offloading. The framework also requires continuous synchronization, which keeps mobile device in active state for whole session of distributed platform.

(*d*) *Mobile cloud execution framework*: In Hung et al. (2012), a mobile cloud-based application execution framework has been proposed. The framework does not demand application redesign.

Unlike a traditional VM-based scheme application migration, the proposed scheme only requires the transfer of application saved states instead of entire states of VM. For migrating an application, the framework first pauses an application on a mobile, sends saved state data files by the application to the cloud, and finally resumes the application execution in cloud server. As the state data files are smaller in size it causes low overhead. To avoid the loss of input data, application replay technique is integrated with already state-saving scheme. The proposed framework provides code security through encryption and isolated execution environment. Moreover, the performance of interactive applications may degrade in low bandwidth networks.

(*e*) *Cloudlet aided cooperative terminals service environment* (*CACTSE*): CACTSE (Qing et al., 2013) is proposed as a cloudlet-based content delivery framework. The framework has a service manager that manages the content distribution among the mobile nodes. The cloudlet does not store the actual data, however, it indexes the contents. The mobile devices register themselves with the service manager and thereafter they request data from the server. The service manager first searches the local index for the requested file. If found, the service manager directs the requesting node to the other node that contains the requested file. If the requested data is not available on the local nodes cache, thereafter, the service manager can then connect to Internet to download the required data file. The objective of CACTSE is to enable the cooperation among local mobile nodes to exchange the data without using Internet or the cloud service. CACTSE pushes the contents closer to the users without disrupting the existing content delivery networks. The framework predicts the future demands based on user's behavior; thereby, the demanding contents can be transmitted to the edge in a proactive manner. However, CACTSE increases the management load in the access network. The cached contents can be outdated with the passage of time.

(*f*) *Cloudlet-based multi-lingual dictionaries*: A cloudlet-based multi-lingual dictionary is proposed in Achanta et al. (2012). The cloudlet-based multi-lingual dictionary leverages the VM-based cloudlet infrastructure to execute the application. The proposed solution aims to minimize the VM migration overhead by employing VM synthesis. The VM synthesis combines two VMs called dictionary base VM and dictionary overlay VM. The dictionary base VM resides within the cloudlet and the dictionary overlay VM resides on the mobile device. The dictionary overlay VM is migrated from the mobile device to the cloud server where the dictionary overlay VM is applied to the dictionary base VM. As a result, a launched VM is derived from the VM synthesis. The dictionary application and voice recognition is executed on launch VM and results are sent back to the mobile device where results are displayed. The proposed solution reduces the transmission overhead and cost. However, the VM distribution in the cloudlet-based multi-lingual dictionaries is coarse-grained.

(*g*) *Dynamic cloudlet*: In Verbelen et al. (2012b), a dynamic cloud approach that deals with applications on component level is presented. The framework does not require the fixed infrastructure near the access point but instead any device in local area network with sufficient available resources can be a cloudlet. In dynamic approach, all devices share their resources. Secondly, as a unit of distribution, VM-based cloudlet uses a coarse-granular approach of VM distribution that has its own deployment complexities. As resources of cloudlet are limited, the latency critical components are executed in cloudlet while non-real components are executed in remote cloud.

The proposed solution provides hierarchical cloudlet deployment on multiple nodes, which enhances scalability of the system. The proposed solution supports the component level application migration instead of whole VM migration. The solution reduces the data transfer size because of only migrating the compute-intensive component. However, the framework has high runtime dynamic application partitioning overhead and complex offloading decisioning process.

(*h*) *Virtual mobile cloud computing* (*VMCC*): VMCC (Huerta-Canepa and Lee, 2010) aims to enhance computing capabilities of stable mobile devices by leveraging an available resources of nearby mobile devices to execute the compute-intensive tasks with low latency and network traffic overhead. During the execution, the application is partitioned into small codes and migrated to nearby mobile devices for execution. The migrated code executes on the nearby mobile devices and reintegrated back upon completion. VMCC reduces the WAN latency by enabling the compute-intensive application execution on local mobile devices that requires no Internet connectivity. Similar to Hyrax,

VMCC also does not consider the mobility. The overall performance of offloading solely depends on number of locally available nodes. Moreover, VMCC requires time-intensive neighbour discovery process.

(*i*) *Mirroring mobile device*: Zhao et al. proposed a mirror server-based framework that provides a VM for mobile devices inside the telecommunication service provider network. The framework shifts some of the workload to the mobile device mirror in the cloud server. The server supports a loose synchronization between the mobile device and the mirror server, thereby, enabling the replays of the inputs to the mobile device on its mirror. The support of data caching on the mirror enables the caching of downloaded files from Internet into mirror cache. The files can be directly retrieved by subsequent users from the server cache instead from Internet server. Similarly, when a user wishes to send a file to multiple users, the mobile device does not need to send a file multiple times on the wireless links. The mobile device just sends the file to the mirror server and mirror server sends to mobile devices on their request. The mirroring of the mobile device reduces the operational overhead on a mobile device by transferring functionality of uploading to the mirror server. The framework puts high synchronization overhead that requires more network bandwidth.

(*j*) *Code offload by migrating execution transparently* (*COMET*): COMET (Gordon et al., 2012) focuses on transparent migration of multi-threaded application to the locally available servers. The framework takes the migration decision considering the workload of machines. COMET takes benefit from the distributed shared memory techniques, such as field level granularity, to maintain the consistency among the end systems. Multiple readers' and writers' threads of application in COMET can simultaneously use the field without any coordination. COMET provides the VM-synchronization between the mobile device and the cloud server. A scheduler in COMET migrates the threads between the end-points to optimize the throughput. The scheduler incorporates the past behavior of thread execution in thread migration decision. The past behavior is captured by monitoring how long a thread has been running on the mobile device without calling a native method. COMET provides transparent migration of partial threads and multi-threads of an application. However, the framework does not provide security mechanism and does not ensure the data integrity.

(*k*) *Replicated application framework*: In Lee (2012), application replication-based framework is proposed that comprises WorkerNodes and MasterNodes. The WorkerNodes run the migrated classes whereas the MasterNodes receive the offloading requests from mobile clients and forward to the appropriate WorkerNodes. To reduce the latency, first request only has an identifier of the class to be migrated, but when a response from cloud server exposes the absence of the replica in the cloud, then the class binary file is migrated from the mobile client to the MasterNode. The framework reduces the application migration overhead by replicating the application inside the cloud for later use. The offloading decisioning is lightweight because the decisioning is performed inside the cloud instead of mobile device. However, the offloading decisioning in the cloud requires information from mobile device such as network connectivity, user preferences, and mobile device capabilities.

(*l*) *Cuckoo*: Kemp et al. (2012) propose a framework, Cuckoo, for partial offloading of mobile applications to the nearby/cloud servers. The Cuckoo aims to facilitate the developers by leveraging the development tools that are known to the developers. The

Table 3
Comparison of bi-objective optimization-based frameworks based on objectives.

Bi-objective optimization-based frameworks	Optimizing execution cost	Minimizing execution time	Reducing offloading time	Reducing network latency	Maximizing resource utilization	Minimizing energy consumption	Optimizing data transfer cost	Optimizing bandwidth utilization	Minimizing VM migration overhead	Minimizing response time
VM-based cloudlet (Satyanarayanan et al., 2009)	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗
AIOLOS (Verbelen et al., 2012a)	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗
Calling the cloud (Giurgiu et al., 2009)	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗
Mobile cloud execution framework (Hung et al., 2012)	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
CACTSE (Qing et al., 2013)	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗
Cloudlet-based multi-lingual dictionaries (Achanta et al., 2012)	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗
Dynamic cloudlets (Verbelen et al., 2012b)	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗
Virtual mobile cloud computing (VMCC) (Huerta-Canepa and Lee, 2010)	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗
Mirroring mobile device (Zhao et al., 2012)	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
COMET (Gordon et al., 2012)	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗
Replicated application framework (Lee, 2012)	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗
Cuckoo (Kemp et al., 2012)	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗
MOCHA (Soyata et al., 2012)	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓

Table 4
General comparison of bi-objective optimization-based frameworks.

Application execution frameworks	Network latency	QoS support	Profiler overhead	Scalable	Programmer support	Cloud usage overhead	Energy consumption	Operational cost
VM-based cloudlet (Satyanarayanan et al., 2009)	Low	n/a	n/a	Yes	n/a	Med.	Low	Med.
AIOLOS (Verbelen et al., 2012a)	Low	No	Low	No	No	Low	Low	Low
Calling the cloud (Giurgiu et al., 2009)	Low	n/a	High	No	No	Low	Low	Low
Mobile cloud execution framework (Hung et al., 2012)	High	Yes	Low	No	Yes	Low	Low	Low
CACTSE (Qing et al., 2013)	Low	Yes	n/a	Yes	No	Low	Low	Low
Cloudlet-based multi-lingual dictionaries (Achanta et al., 2012)	Low	n/a	n/a	Yes	n/a	Med.	Low	Low
Dynamic cloudlets (Verbelen et al., 2012b)	Low	Yes	High	Yes	Yes	Low	Low	Low
Virtual mobile cloud computing (VMCC) (Huerta-Canepa and Lee, 2010)	Low	No	n/a	No	No	High	Low	Low
Mirroring mobile device (Zhao et al., 2012)	Low	n/a	n/a	Yes	n/a	High	Low	High
COMET (Gordon et al., 2012)	Low	n/a	n/a	Yes	Less	Low	Low	Low
Replicated application framework (Lee, 2012)	Low	No	Yes	Yes	n/a	High	Low	High
Cuckoo (Kemp et al., 2012)	Low	n/a	n/a	Yes	n/a	Low	Low	Low
MOCHA (Soyata et al., 2012)	High	Yes	n/a	Yes	n/a	Med.	Med.	High

Med. – Medium, n/a – Not Applicable.

Cuckoo consists of two type of builders: Cuckoo Service Rewriter (CSR) and Cuckoo Remote Service Deriver (CRSD). The Cuckoo separates computation intensive (services) and interactive components (activities) of the application using the Android Interface Definition Language (AIDL) (2012). Once the service is available on server, the address of the server is sent to the resource manager running on the mobile device. Then, address registrar registers the server address to enable the resources usable for the mobile device. The Cuckoo framework requires programmers support for application modification. The Cuckoo takes static decisions for offloading that are context unaware. Moreover, the Cuckoo provides proxy-based application execution that incurs additional delay.

(*m*) *Mobile–cloudlet–cloud acceleration architecture* (*MOCHA*): MOCHA is proposed in Soyata et al. (2012) to leverage on the mobile proximate cloudlet resources. MOCHA architecture consists of three components: mobile device, cloudlet, and the cloud. The mobile device is connected to the cloudlet which, in turn, is connected to a larger cloud infrastructure such as Amazon Web Services, Windows Azure. The cloudlet finds how to partition the computation among the cloudlet and multiple servers in the cloud based on different QoS metrics. MOCHA aims to minimize the latency from user to the cloud. MOCHA supports parallel processing for application execution and provides QoS. However, acquiring the network QoS parameters and finding the best QoS path consume mobile device energy.

Table 3 presents the comparative summary of bi-objective optimization-based frameworks considering the objective. The general comparison of these frameworks is presented in Table 4.

4.3. Multi-objective optimization-based frameworks

The multi-objective optimization-based frameworks mainly focus on optimizing the multiple objectives of the frameworks. The multi-objective optimization-based frameworks have highest complexity than other categories, mono-objective optimization-based frameworks and bi-objective optimization-based frameworks. These frameworks incorporate a variety of parameters in the objective function. The multi-objective optimization-based frameworks are better than others in diverse running environment and to fulfill the requirements of different applications in MCC. Herein, we are presenting the frameworks that aim to optimize multiple objective functions:

(*a*) *Elastic application model*: In Zhang et al. (2010), an optimal elastic application model is presented, which enables the use of cloud resources in a transparent manner. The proposed elastic framework incorporates four attributes in the cost model: minimizing power consumption, minimizing monetary cost, maximizing throughput, and maximizing security and privacy. The application is partitioned into various components called weblets that are replicated across multiple clouds; thereby, enhancing the availability and reliability. Moreover, the application execution configuration is optimized by the following objective function:

$$y^* = \arg\max_{y} p(y) \prod_{i=1}^{L} p(x_i|y) \prod_{j=1}^{M} p(z_j|y) \tag{14}$$

The vector '*x*' has values of different device status components such as throughput, memory usage, upload bandwidth, and file cache. The vector '*z*' has the values for user's preferred options such as processing speed and monetary cost. The variable '*y*' is the configuration variable representing the total number of configurations. The variables '*L*' and '*M*' represent the number of components in the status vector and the number of components in the preference vector, respectively. The establishment of runtime distributed platform and its management require additional computing resources. The offloading decision is based on complex cost model which incorporates a number of parameters; thereby making the offloading decision process compute-intensive.

(*b*) *ThinkAir*: ThinkAir (Kosta et al., 2012) exploits mobile device virtualization to simultaneously execute multiple offloaded methods; thereby, reducing the application execution time. The remote method invocation is usually done through execution controller, which resides on mobile device as well as on the cloud side. The execution controller takes the offloading decision based on the information collected by the profiler. If the remote connection fails, the framework falls back to local execution. The cloud side of offloaded code is managed by application server. The framework supports six types of VMs with different CPU, memory, and heap sizes. The default server is called primary server, which always stays online and the rest are of secondary server type. Moreover, the framework supports accurate and lightweight profiling for hardware, software, and network. The framework supports the parallel execution and considers the multi-users application offloading scenarios for execution in the cloud that is more realistic. However, the offloading decisioning process of ThinkAir is complex. The framework also requires the installation of android operating system-based VM on the cloud server.

(*c*) *Pocket cloudlet*: A storage-based cloudlet is proposed in Koukoumidis et al. (2012) that is aiming to utilize the large

Table 5
Comparison of multi-objective optimization-based frameworks based on objectives.

Multi-objective optimization-based frameworks	Minimizing energy consumption	Maximizing throughput	Minimizing monetary cost	Maximizing security & privacy	Minimizing execution time	Minimizing execution cost	Maximizing Cache Hit Rate	Reducing network latency	Minimizing missed deadlines	Easing application development	Minimizing data exchange
Elastic application model (Zhang et al., 2010)	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗
ThinkAir (Kosta et al., 2012)	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗
Pocket cloudlet (Koukoumidis et al., 2012)	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗
MISCO (Dou et al., 2010)	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗
XMPP-based mobile cloud middleware (Kovachev et al., 2012a)	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓
Mobile augmentation cloud services (MACS) (Kovachev et al., 2012b)	✓	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓

available storage capacity of the nearby mobile devices to reduce the latency and energy issues in accessing the distant cloud services. The framework leverages on both the community access model and personal access model to maximize the hit rate of user queries. The partial or full cloud services are migrated to a mobile device; thereby, transforming a mobile device into a pocket cloudlet. The pocket cloudlet improves the user experience by implementing the following functionalities: (a) caching the information on proximate mobile devices; (b) personalizing the service based on behavior and usage patterns of the individual users; and (c) ensuring the privacy of individual users by locally running the service on the mobile device. The pocket cloudlet improves the mobile user experience by providing instant access to the information. The pocket cloudlet reduces overall service discovery time and energy consumption. However, the pocket cloudlet requires real-time updates over the radio links to ensure the cached data freshness.

(*d*) *MISCO*: MISCO (Dou et al., 2010) extends the MapReduce to the distributed cloud environment that is composed of mobile worker nodes and centralized server. The MapReduce is a data processing framework that enables the parallel execution of an application in cluster environment. A MapReduce framework is implemented on a centralized master server. A unique feature of the framework is that mobile devices are the worker nodes, which provide remote application processing. The worker nodes get the workload from the master server, perform computations, and return the results to the master server. MISCO employs a centralized monitoring mechanism for monitoring of the distributed execution platform. MISCO provides powerful programming abstract and supports parallel processing. However, the mobile devices have high transmission overhead for input, synchronization, monitoring, and results exchange.

(*e*) *Extensible messaging and presence protocol (XMPP)-based mobile cloud middle-ware*: A XMPP-based mobile cloud middleware is presented in Kovachev et al. (2012a) that provides application partitioning and adaptive offloading to available nearby server. The proposed middle-ware covers two crucial aspects of mobile cloud computing architecture: a context-aware cost model and standardization of XMPP as a cloudlet protocol. The offloading decision is based on context-aware cost model, which comprises different parameters particularly module execution time, battery level, resource consumption, security, monetary cost, and network bandwidth. The objective of the model is to perform intelligent decisions with minimal overhead while satisfying the environment constraints. The optimization model in XMPP is supported with a lightweight, efficient and predictive decision algorithms. The XMPP reduces the WAN latency, transmission delay, incorporates QoS, but suffers with profiling overhead. The offloading is lightweight as only the prediction algorithm runs on the mobile device whereas optimization algorithm runs in the cloudlet. However, the middle-ware requires programmer support to annotate the methods for remote execution in MCC.

(*f*) *MACS*: An adaptive middleware which provides lightweight application partitioning, seamless computational offloading, and resources monitoring is presented in Kovachev et al. (2012b). The partitioning decision is transformed into optimization problem and solved by optimization solver. The cost function in optimization problem comprises transfer cost of service, its related services which are not collocated, CPU cost, and memory cost of mobile device. The constraints are related with minimization of memory usage, energy usage, and execution time. The cost function is represented as follows:

$$\min_{x \in 0,1} (c_{transfer} * w_{tr} + c_{memory} * w_{mem} + c_{CPU} * w_{CPU}) \quad (15)$$

Table 6
General comparison of multi-objective optimization-based frameworks.

Application execution frameworks	Network latency	QoS support	Profiler overhead	Scalable	Programmer support	Cloud usage overhead	Energy consumption	Operational cost
Elastic application model (Zhang et al., 2010)	High	Yes	High	Yes	Yes	High	Low	High
ThinkAir (Kosta et al., 2012)	n/a	No	High	Yes	Yes	High	Low	Low
Pocket Cloudlet (Koukoumidis et al., 2012)	Low	No	Med.	Yes	n/a	Low	Low	Low
MISCO (Dou et al., 2010)	Low	No	High	Yes	n/a	Low	High	Low
XMPP-based mobile cloud middleware (Kovachev et al., 2012a)	Med.	Yes	High	Yes	Yes	Low	Low	Med.
MACS (Kovachev et al., 2012b)	Low	No	High	No	Yes	Low	Low	Low

Med. – Medium, n/a – Not Applicable.

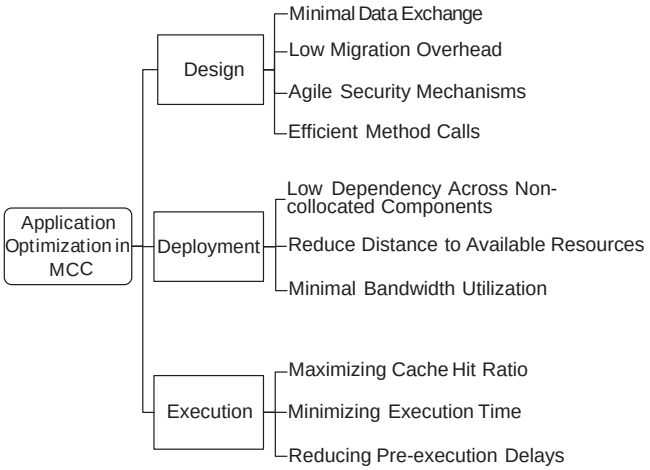

Fig. 2. Taxonomy of application optimization strategies in MCC.

where

$$c_{transfer} = \sum_{i=1}^{n} \text{code}_i * x_i + \sum_{i=1}^{n} \sum_{j=1}^{k} \text{tr}_j * (x_j \text{ XOR } x_i) \quad (16)$$

$$c_{memory} = \sum_{i=1}^{n} \text{mem}_i * (1 - x_i) \quad (17)$$

$$c_{CPU} = \sum_{i=1}^{n} \text{code}_i * \alpha * (1 - x_i) \quad (18)$$

In above equations, n is number of offloaded modules. For a particular module i, the code size and memory cost are represented by code_i and mem_i, respectively. The transfer size of module i is represented by tr_i that is equal to sum of send_i and rec_i. The x_i is a decision variable that shows whether the module i is executed locally ($x_i = 0$) or remotely ($x_i = 1$). Although MACS is lightweight and provides dynamic partitioning and seamless offloading, it requires in advance developer support for structuring the code in a model. Moreover, the offloading process is affected with profiling, partitioning, and migration overhead.

Table 5 presents the comparative summary of multi-objective optimization-based frameworks considering the objective. The general comparison of these frameworks is presented in Table 6.

5. Cloud-based mobile application-centric taxonomies

In this section, we present the three taxonomies related to cloud-based mobile application in MCC. First taxonomy classifies the optimization strategies that are applied to optimize the application performance. Second taxonomy classifies the operations involved during application execution in MCC. Third taxonomy classifies various attributes of application execution frameworks.

5.1. Taxonomy of application optimization strategies in MCC

The application in MCC can be optimized with respect to three aspects: (a) design, (b) deployment, and (c) execution. Figure 2 presents the taxonomy of application related optimization strategies in MCC. The application design can be optimized by employing following features: (a) minimizing data exchange, (b) employing lightweight migration framework, (c) leveraging agile security mechanisms, and (d) alleviating method call overhead. The data exchange can be minimized by reducing the component dependency in the design of an application. The lightweight frameworks can be designed by either reducing the runtime overhead or shifting non-interactive functionalities from mobile device to cloud side. The agile security mechanisms can be employed at the design of the application by either relying on cloud side security such as reputation-based trust (Satyanarayanan et al., 2009) or using low overhead ciphers (Goyal and Carter, 2004). The method call overhead can be reduced either by using callback function (Verbelen et al., 2012a) or by reducing the number of method calls that can be achieved by implementing multiple similar operations in a single method (Cuervo et al., 2010).

The deployment of application can be optimized by efficiently deploying the components in the execution environment. The efficient application deployment can be attained by considering the following concerns: (a) the dependency across the non-collocated components is required to be minimized (Zhang et al., 2010), (b) the distance to the available resources should be reduced, (c) the bandwidth utilization should be minimal, and (d) the runtime cost should be lower. The following type of components should run on the mobile device: (a) user interface (Giurgiu et al., 2009), (b) the component having a hardware dependency on the mobile device (Cuervo et al., 2010), (c) real time components (Verbelen et al., 2012b), and (d) components that require secure data (Zhang et al., 2010). The execution of application in MCC can be optimized by employing the following features in the execution environment: (a) caching (Satyanarayanan et al., 2009; Fesehaye et al., 2012), (b) parallel execution (Zhang et al., 2010; Kosta et al., 2012), and (c) pre-installations (Goyal and Carter, 2004; Lee, 2012). The caching can help in reducing the application response time by pre-fetching the data and storing it locally for subsequent use. The parallel execution reduces the overall execution time in the cloud but it requires effective task scheduling. The pre-installation minimizes the pre-execution delay in MCC.

5.2. Taxonomy of application execution operations in MCC

This section presents the taxonomy of operations performed by application execution frameworks in MCC. Figure 3 presents the taxonomy of application execution operations in MCC. The

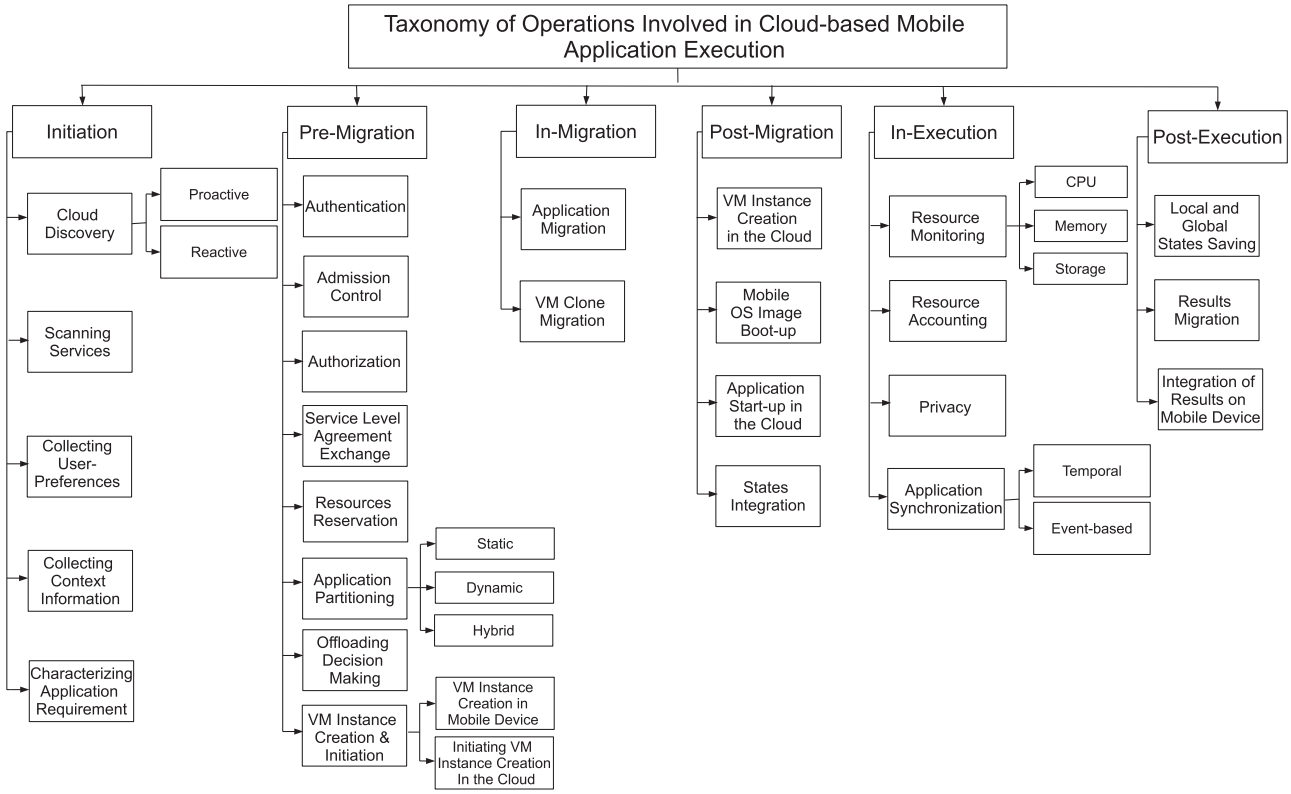

Fig. 3. Taxonomy of application execution operations in MCC.

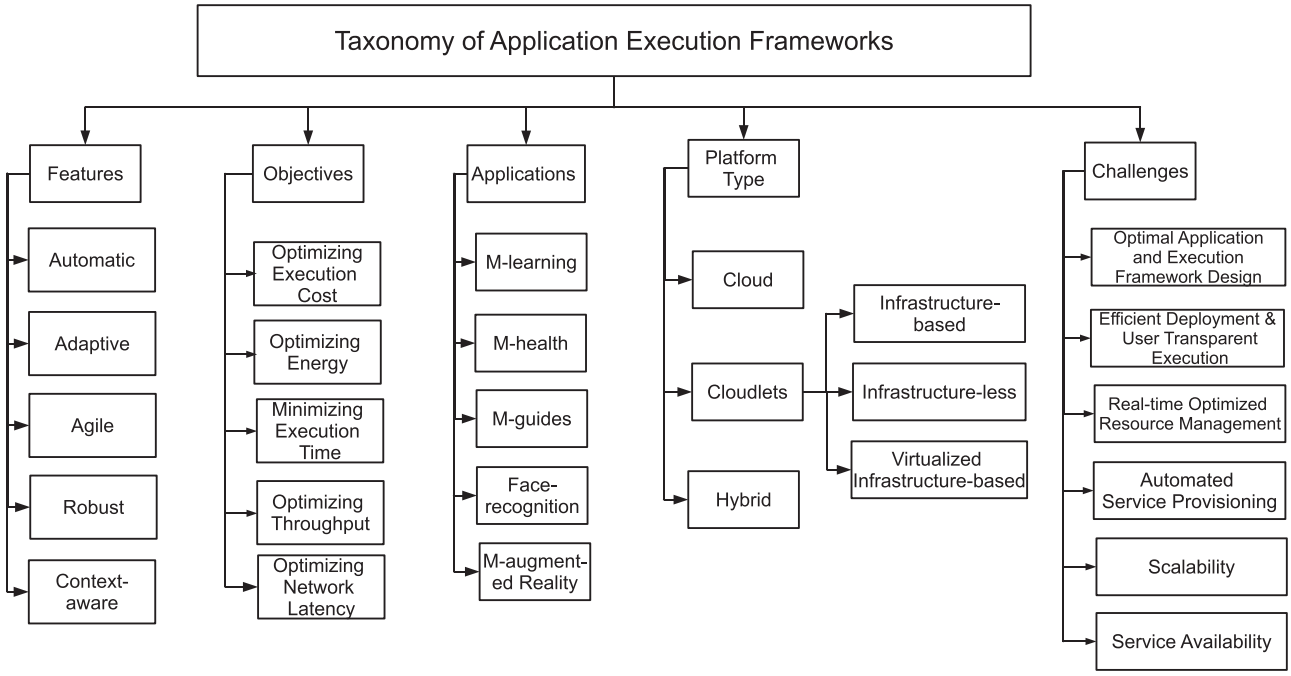

Fig. 4. Taxonomy of application execution frameworks.

operations involved in cloud-based application execution are classified into five categories: initiation, pre-migration, in-migration, post-migration, in-execution, and post-execution.

Initiation operations are related to cloud discovery, scanning of services, collecting user-preferences, gathering context information, and characterizing application requirements. The cloud discovery is the process of finding out the cloud server for leveraging the available resources on the server to execute the application. The cloud discovery can be performed in two ways either in a pro-active manner or in a reactive manner. The pro-active cloud discovery can reduce the pre-execution delay whereas reactive cloud discovery induces delay before the start of actual execution. The scanning of services discovers the services offered by a cloud. The service scanning is performed on user request. Therefore, the

service scanning operation is also time-intensive task. The user-preference collection operation gets the user preferences on cost, quality, and other performance metrics. The device and network-related context information is also collected in the initiation operation. The context information include the device current workload and residual battery power. The application requirements are also collected and characterized during the initiation of application.

The pre-migration involves in authentication, admission control, authorization, service level agreement exchange, resource reservation, application partitioning, offloading decisioning, VM instance creation and initiation. Authentication verifies that someone is who he claims, he actually is. Admission control is a validation process that checks whether the cloud resources are sufficient for the requested application execution before the start of the actual execution. The authorization is the process of specifying access rights to cloud resources. The service level agreement is exchanged between mobile users and cloud service provider to exchange the resource requirements and the service policies. The resource reservation process reserves the resources of cloud server for mobile user as agreed in service level agreement. The application partitioning is process that involves in dividing the application in transferable and non-transferable components. The application partitioning is performed in static, dynamic, and hybrid form. Another important operation that is performed in pre-migration phase is offloading decision making. The offloading decision making is performed considering the runtime conditions of environment. The VM instance creation and migration is the last operation in pre-migration phase, which creates VM instance in mobile device and VM instance creation in the cloud is initiated.

The in-migration operations deal with the migration of application, its running states, and the VM clone migration. The migration operation is dependent on the underlying wireless network technologies that can be WiFi or cellular technologies. The post-migration operations involved in VM instance creation in the cloud, mobile OS image boot-up on the cloud server, application start-up in the cloud, and running states of migrated application in the cloud. The post-migration phase actually setups the platform for execution of mobile application in the cloud. The in-execution phase involves in resource monitoring in the cloud. The resources are CPU, memory, and storage. The resources accounting is also performed to systematize information about resources utilization in the cloud. The privacy is also ensured during the in-execution phase of the application. Lastly, the application components are synchronized during the in-execution phase. The application synchronization is of two types: (a) temporal and (b) event-based. The temporal synchronization is performed periodically whereas event-based application synchronization is performed on trigger of an event. The post-execution phase involves local and global states saving, result migration, and integration of results on mobile device. The local and global states of an application are saved in case if the computation migration is required. Otherwise just results are saved and pushed back to the mobile device.

5.3. Taxonomy of application execution frameworks

This section presents the taxonomy of application execution frameworks as shown in Fig. 4. The features attribute of the taxonomy presents the characteristics of the optimized application execution frameworks. The features attribute shows that the framework is required to be automatic, user transparent, adaptive, agile, robust, and context-aware to attain the optimized application execution.

The frameworks are required to support automated partitioning, offloading, and automated cloud-based service provisioning mechanisms to attain the optimal application execution. User transparency can be achieved by minimizing the user involvement in the execution process. An application execution framework is required to implement two functionalities to efficiently adapt according to the changes in the environment: (a) change detection mechanism, (b) response mechanism. The change detection mechanism is required to be accurate, user transparent, and agile. The response mechanism should peacefully degrade the quality of application and respond to the resources in non-greedy opportunistic way. The agile feature can be attained by minimizing the overhead involved and expediting the computation process. The robustness allows the framework to execute the application even if the failure occurs. Context-awareness allows the framework to know the available opportunities in terms of resources and services in mobile cloud computing.

The state-of-the-art application execution frameworks are designed to address different objectives such as optimizing execution cost, minimizing energy consumption, maximizing throughput, minimizing execution time, and minimizing network latency. The taxonomy also presents some of the cloud-based mobile applications such as m-learning, m-health, m-guides, face-recognition and m-augmented reality. The application execution frameworks can also be categorized on the basis of deployment platform type. The framework deployment is of three different types: (a) cloud, (b) cloudlet, and (c) hybrid. Cloud provide remote resources and services. The application execution in the cloud suffers from high WAN latency which obstructs the realization of the vision of optimal application execution. The cloudlets are of three types: (a) infrastructure-based, (b) infrastructure-less, and (c) virtualized infrastructure-based. The infrastructure-based cloudlet requires a deployment of server in WLAN or in telecommunication service provider network. The infrastructure-less cloudlet does not require any server for the deployment of cloudlet, instead it uses the resources of available mobile devices. The virtualized infrastructure-based cloudlet performs component level application migration instead of whole VM. Some of the components use mobile device resources while the rest utilizes the cloud resources. Hybrid platforms are formed by integration of mobile ad-hoc cloudlet, local cloudlet, and the remote cloud. The taxonomy also presents the challenges in realizing the vision of near-to-optimal application execution in MCC.

6. Metrics for optimal application migration in MCC

This section discusses the suitability of various metrics for optimal application migration of delay-sensitive mobile applications. Figure 5 shows the metrics; the investigation of these metrics in context of optimal application execution is an important research perspective. These metrics belong to five areas namely *mobile device*, *network*, *application type*, *user preferences*, and *cost*. The significance of each metric with respect to optimal application execution in MCC is discussed herein.

6.1. Mobile device

Mobile device related metrics mainly exploit device capabilities in terms of processing load, CPU speed, memory, storage, wireless access technologies, and number of interfaces. The processing load represents the average CPU workload, while available memory and available storage provide the information of idle memory and storage, which can be used by the application. If processing load is higher, then new application process may not have sufficient number of CPU cycles for its execution. Hence, the application requires migration to nearby or remote cloud server for smooth execution. The available memory and storage provide memory

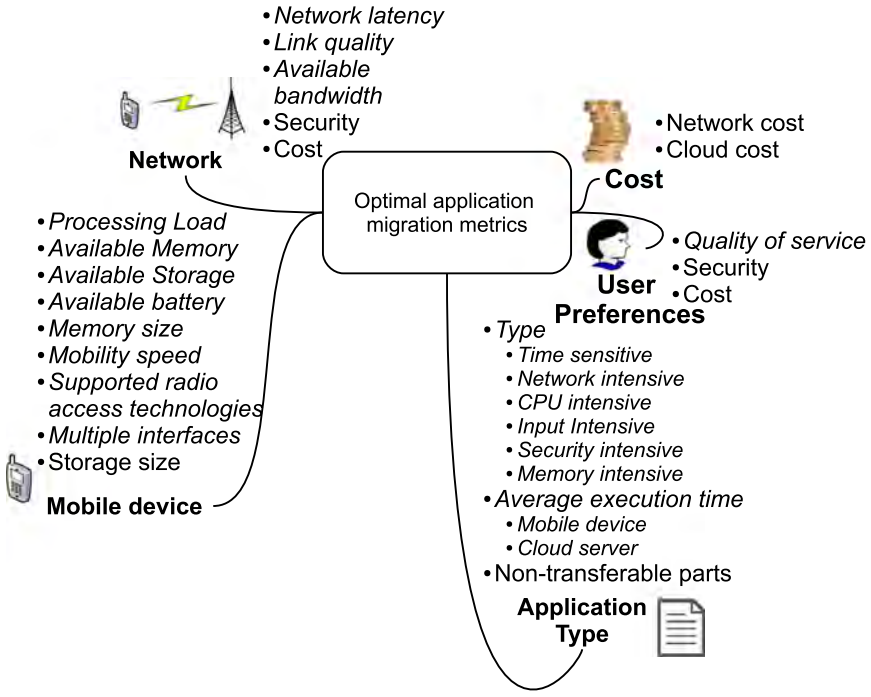

Fig. 5. Metrics for optimal application migration in MCC.

space for intermediate application states and results. The low memory disrupts the execution of the application and low storage restrains the device from saving results. Hence, low memory and storage impede the smooth execution of the application and in-turn deteriorates the user experience. The mobility pattern of mobile devices also affects the link failure across the mobile network (Lenders et al., 2006). The offloading decision also requires to incorporate the mobility pattern. The access technology that is used for offloading an application also affects the application performance in terms of latency and packet delivery ratio which also in-turn degrades the user experience. Furthermore, available battery power also disrupts the execution of mobile application. The mobile device with low battery power can complete execution of the CPU-intensive application by offloading into the cloud while conserving the battery power. Nowadays, mobile devices support multiple interfaces so offloading can be improved by exploiting available multiple interfaces to implement bandwidth aggregation. Bandwidth aggregation helps in improving the throughput, packet delivery ratio, and reliability. Furthermore, bandwidth aggregation can increase the capacity for application offloading (Ramaboli et al., 2012).

6.2. Application type

The characteristics of mobile applications vary from application to application as each application serves different needs of users. The offloading decision is influenced by application type such as if an application is more bandwidth-intensive than that of its computational requirement, then it is wise decision to execute application locally at mobile device (Hung et al., 2012). The work in Zhang and Figueiredo (2006) classifies the mobile applications as CPU-intensive, memory-intensive, and input/output-intensive. The mobile applications can also be delay-sensitive (Nazir et al., 2009), security-intensive (Cano and Domenech-Asensi, 2011; Khan et al., 2013; Sookhak et al., 2014), and network-intensive (Ballagas et al., 2007). Memory- and CPU-intensive applications are suitable candidates for offloading; whereas network, input and security-intensive applications have constraints to run in the remote cloud. However, offloading decision of delay-sensitive application is based on network latency and on execution time. If local execution time of an application is longer than its remote execution time, then offloading is beneficial (Verbelen et al., 2012a). In some cases, offloading entire application degrades the application performance. Such types of applications have two types of components namely transferable and non-transferable parts. The non-transferable parts can be one of the following: (1) a module that involves in input/output (Cuervo et al., 2010; Othman and Hailes, 1998; Ou et al., 2006a); (2) a module that interacts with mobile device hardware directly (Othman and Hailes, 1998); (3) a module that depends on native code (Gu et al., 2004) or on the module of the application which resides on mobile device (Ou et al., 2006b); and (4) a module that uses directly device related information (Gu et al., 2004). In short, to achieve the optimal application execution in mobile cloud computing environment, the tasks are required to be executed in the most suitable location considering the characteristics of the task.

6.3. User preferences

Mobile applications run in dynamic wireless environment with a variety of access technologies, where tradeoffs exist between various available options such as high bandwidth link and cost. End user prioritizes the available options to achieve the optimal execution of application while satisfying all constraints related to QoS, cost, and security. QoS related preferences are set by user to attain the sufficient level of Quality of Experience considering his budget. There exists a direct proportionality between QoS and monetary cost, but usually user demands high QoS in low monetary cost. The service providers charge different rates for different levels of QoS. Hence, the QoS is an important parameter for optimal application execution of a mobile cloud application. The QoS improves overall latency and throughput to alleviate the performance gap between local and remote execution. Similar to QoS, security also increases the user monetary cost. Furthermore, it also increases the processing complexity and communication overhead for application execution. The increase in processing

complexity and communication overhead put additional delay, which impedes the realization of optimal application execution in MCC. Hence, security related features specifically authentication and encryption are required to be wisely enabled considering the application and end user requirements. The cost varies over the range of services and for different types of networks. The cost related options facilitate a user in selection of different services considering his/her budget. For application requiring optimal execution, a user can pay more to get higher bandwidth share and obtain priority for his/her traffic in network.

6.4. Cost

The cost mainly involves monetary cost of wireless networks and the cloud. The wireless networks such as 3G links charges more to mobile users whereas WiFi is freely available access technology. User can use WiFi for application offloading and reintegration of results to reduce his overall cost. Whenever WiFi is not available, a mobile user can switch to 3G for offloading his application into the cloud. Different cloud service providers charge the same service on different rates. User selects the cloud among the available clouds considering the rate charged by the service provider. The cost varies for different amount of resources so a user can buy more resources to expedite the execution process by performing parallel executions. To attain the optimal application execution, user can pay more charges to access and utilize more resources.

6.5. Network

Network related parameters are network latency, wireless link quality, available bandwidth, security, and network cost. Incorporation of these parameters in offloading decision reduces the latency and provides the secure offloading on affordable cost. Selection of network with low latency, with better wireless link quality, and available bandwidth helps in attaining the optimal execution of mobile cloud applications by improving throughput and reducing execution time.

7. Future research directions

The goal of attaining the optimal application execution in MCC opens the doors for research in different dimensions. These research dimensions cover the optimal application designing, cost-effective application execution framework designing, devising and implementing lightweight context information exchange mechanisms, solutions for efficient component deployment, enabling the user-transparent execution, real-time optimized management of heterogeneous environment, automated service provisioning, and scalability of the cloudlet-based solutions.

The mobile applications have to run on the resource-constrained mobile devices, therefore, the applications should be designed in such a way that can reduce the energy consumption, attain better performance and require less resources. The interdependency among distributed modules of an application should be considered at the time of designing. Moreover, the design of application execution frameworks that are responsible to execute the applications in MCC is also important to efficiently utilize the limited resources of mobile devices. The application execution frameworks should be designed considering the following key points: (a) the dependency across the non-collocated components is required to be minimized, (b) the bandwidth utilization should be minimized, (c) the framework should support fault tolerance, and (d) runtime cost should be minimized. Moreover, the high variability in bandwidth availability influences the way execution framework decides and chooses the execution location. The optimization of the application and execution frameworks behavior using application and context aware techniques becomes more important.

The lightweight context information collection mechanisms are required to collect the information related to mobile device and the cloud server. The MCC environment is highly dynamic, therefore, the context information is also required to be frequently collected. Moreover, the efficient and agile mechanisms are required to apply context information to take the offloading decisions in MCC. Moreover, a mobile device can use context information to (a) increase the precision of information retrieval, (b) discover services, (c) adapt interfaces, or (d) make the user interaction implicit.

Another research direction for attaining the optimal application execution is finding the effective component deployment mechanisms. The effective component deployment refers to the deployment of components strategy by which the components can be executed with less computation and communication overhead. The cloud-based mobile applications have multiple components and these components may vary based on their characteristics and functionalities. One of the components is user interface that is involved in input, therefore, the component should be executed on the mobile device. The component that interacts with the hardware of mobile device should also be executed on the mobile device. The components that have dependency on each other should be executed on the same device and dependency among non-collocated components should also be minimized. Therefore, there is a need to investigate the lightweight methods that can find the dependencies among the components on runtime with less overhead on the mobile device.

The mobile application in MCC can run on any of the platform, mobile device, cloudlet or a cloud, based on the availability of resources in local proximity or access to the cloud. The running application can be migrated from mobile device to the cloudlet or the cloud server at runtime. The migration of the application from mobile device to the cloudlet or the cloud is required to be user transparent, therefore, there is a need to design application execution frameworks that can provide user-transparent execution of the application across the platform.

The real-time management of heterogeneous environment in MCC refers to an optimal design, efficient monitoring, and effective maintenance of collaborative environment for execution of the application. Such a vital necessity is of paramount significance in the optimized execution of the application in MCC. The management of computing environment in MCC usually involves the monitoring of the physical assets, such as servers, virtual assets such as VMs, network infrastructure such as control plane of switches and routers, and softwares (Kant, 2009). It also demands efficient and effective deployment of mobility management strategies (Zekri et al., 2012). Therefore, there is a need of designing efficient monitoring and effective maintenance mechanism that can monitor the collaborative cloud environment with less overhead.

The automated service provisioning refers to the provisioning of a service on demand with minimal service provider involvement. Hence, the automated service provisioning requires modeling of application performance prediction model, periodically predicting future demands, and allocating the resources. There is a need to device a model that can predict the application performance, estimate future demands, and allocate resources to the cloud users considering their applications requirements.

8. Open challenges

In this section, we highlight some of the most important challenges that impede the optimization of mobile application in MCC. The discussion on the open challenges provides research

directions to researchers in the domain for further investigations in MCC.

8.1. Optimal application and execution framework design

The optimal design of an application and execution framework involves minimal data exchange, less method call overhead, light-weight migration process, and employment of agile security mechanisms. The complexity of migration process is high because of diverse MCC environment and dynamic conditions of wireless networks. The designing of optimal application and execution framework that can provide the optimal application execution for such diverse and dynamic environment is an open research challenge.

8.2. Efficient deployment and user-transparent execution

The efficient deployment of application involves minimal dependency across non-collocated components, reducing the distance to the available resources, minimal bandwidth utilization, and low runtime cost. However, the efficient deployment of application in MCC is difficult to attain because of the complexity involved in determining the dependency across non-collocated components at runtime. Moreover, attaining the user-transparent application execution in MCC is an open research challenge because of complexity involved in migration process, intrinsic limitations of wireless medium, and heterogeneity across MCC platforms.

8.3. Realtime optimized management of heterogeneous computing environment

The realtime management of the heterogeneous computing environment becomes a challenge due to diverse nature of environmental elements, intrinsic limitations of wireless medium, and the highly dynamic mobility patterns. However, the researchers can obtain guidelines from the efforts already done in a similar domain (Patouni et al., 2012) to design the lightweight solutions for the real-time management of heterogeneous resources. Reducing the number of monitoring elements and shrinking/compressing the amount of information exchange for resource management can aid in attaining the realtime management strategy.

8.4. Automated service provisioning

The goal of automated service provisioning is to provide resources on-demand with minimal service provider involvement. The automated service provisioning is challenging task due to complexity involved in modeling of application performance prediction model and periodically predicting future demands. The automated service provisioning can be realized by integrating lightweight artificial intelligence approaches to predict the application performance and future demands.

8.5. Scalability

The scalability ensures the service provisioning regardless of the number of devices using the services. Scalability of services and resource provisioning become critical for MCC when cloud is not accessible and only limited local resources are available. With limited resources and services, ensuring scalability is a challenging task which affects the smooth execution of an application. In this scenario, the scalability can be ensured by running only the essential background tasks on the mobile device and prioritizing the allocation of resources to the tasks.

8.6. Availability of services

The availability of services in the cloud ensures the provision of remote services. The availability of cloud services in MCC mainly relies on three factors: (1) wireless access medium, (2) residual server capacity, and (3) access latency to the data. The MCC-based execution frameworks face challenges in ensuring the availability of services due to intrinsic limitations of underlying wireless technologies and highly dynamic, non-deterministic server workload. Failure in obtaining sufficient resources for demands of application disrupts the execution of an application. Availability of resources can be ensured by pro-actively managing the local resources to run the application locally in case of network connectivity loss.

9. Conclusions

The paper investigates the state-of-the-art application optimization strategies by conducting survey on the literature. The contribution of the paper is manifold: (a) comprehensive literature survey, (b) comparison of state-of-the-art application execution frameworks, (c) identification of application optimization strategies for attaining the optimal application execution, (d) devisal of application related taxonomies, (e) investigation of metrics suitability for optimal application execution, and (f) highlighting the open research challenges in attaining the optimal application execution. The comparison highlights the similarities and differences of the current frameworks and evaluates the suitability of existing frameworks for optimal application execution.

Some of the current application execution frameworks improve the response time of application for MCC by reducing the WAN latency. A number of current application execution frameworks lack with scalability features and QoS support or suffer from profiling overhead. Besides, some of the current frameworks either demand programmer support or rely on intensive VM-based application deployment. There is need to design optimal metrics for attaining the optimal application execution while considering user preferences and environment constraints. Furthermore, the research is required to be done in the domain of framework designing considering the identified demerits of existing frameworks.

Acknowledgements

This work is supported in part by the Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant (UM.C/625/1/HIR/MOE/FCSIT/03) and by the Bright Spark Unit, University of Malaya, Malaysia.

References

Abolfazli S, Sanaei Z, Ahmed E, Gani A, Buyya R. Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. IEEE Commun Surv Tutorials 2013; 16(1). http://dx.doi.org/10.1109/SURV.2013.070813.00285.

Achanta V, Sureshbabu N, Thomas V, Sahitya M, Rao S. Cloudlet-based multi-lingual dictionaries. In: Proceedings of third international conference on services in emerging markets (ICSEM'03). Mysore, India; 2012. p. 30–6. http://dx.doi.org/10.1109/ICSEM.2012.12.

Ahmed E, Khan S, Yaqoob I, Gani A, Salim F. Multi-objective optimization model for seamless application execution in mobile cloud computing. In: Proceedings of 5th international conference on information communication technologies (ICICT'13). Karachi, Pakistan; 2013. p. 1–6. http://dx.doi.org/10.1109/ICICT.2013.6732790.

Ahmed E, Akhunzada A, Whaiduzzaman M, Gani A, Ab Hamid SH, Buyya R. Network-centric performance analysis of runtime application migration in mobile cloud computing. Simul Model Pract Theory 2015a [in press].

Ahmed E, Gani A, Abolfazli S, Yao L, Khan S. Channel assignment algorithms in cognitive radio networks: Taxonomy, open issues, and challenges. IEEE Commun Surv Tutor 2015b [in press].

Ahmed E, Qadir J, Baig A. High-throughput transmission-quality-aware broadcast routing in cognitive radio networks. Wirel Netw 2015c [in press].

Android interface definition language (AIDL) [online], ⟨http://developer.android.com/guide/developing/tools/aidl.html [accessed December 2012]. ⟩.

Ballagas R, Kratz S, Borchers J, Yu E, Walz S, Fuhr C, et al. Rexplorer: a mobile, pervasive spell-casting game for tourists. In: CHI07' extended abstracts on human factors in computing systems. San Jose, California, USA: ACM; 2007. p. 1929–34.

Cano M, Domenech-Asensi G. A secure energy-efficient m-banking application for mobile devices. J Syst Softw 2011;84(11):1899–909.

Chin WH, Fan Z, Haines R. Emerging technologies and research challenges for 5G wireless networks. IEEE Wirel Commun 2014;21(2):106–12.

Chun B, Maniatis P. Augmented smartphone applications through clone cloud execution. In: Proceedings of the 8th workshop on hot topics in operating systems (HotOS), Monte Verita, Switzerland, vol. 9; 2009. p. 8–11.

Chun B, Ihm S, Maniatis P, Naik M, Patti A. Clonecloud: elastic execution between mobile device and cloud. In: Proceedings of the sixth conference on computer systems; 2011. p. 301–14.

Cruz D, Barros E. Vital signs remote management system for PDAs. In: Proceedings of 8th Euromicro conference on digital system design. Porto, Portugal: IEEE; 2005. p. 170–3.

Cuervo E, Balasubramanian A, Cho D, Wolman A, Saroiu S, Chandra R, et al. MAUI: making smartphones last longer with code offload. In: Proceedings of the 8th international conference on mobile systems, applications, and services. San Francisco, CA, USA: ACM; 2010. p. 49–62.

Dinh HT, Lee C, Niyato D, Wang P. A survey of mobile cloud computing: architecture, applications, and approaches. Wirel Commun Mob Comput 2013;13(18):1587–611.

Dou A, Kalogeraki V, Gunopulos D, Mielikainen T, Tuulos V. MISCO: a mapreduce framework for mobile systems. In: Proceedings of the 3rd international conference on PErvasive Technologies Related to Assistive environments (PETRA'03). Samos, Greece: ACM; 2010. p. 32.

Fernando N, Loke S, Rahayu W. Mobile cloud computing: a survey. Future Gener Comput Syst 2013;29(1):84–106.

Fesehaye D, Gao Y, Nahrstedt K, Wang G. Impact of cloudlets on interactive mobile cloud applications. In: 16th IEEE international enterprise distributed object computing conference. Beijing, China: IEEE; 2012. p. 123–32.

Giurgiu I, Riva O, Juric D, Krivulev I, Alonso G. Calling the cloud: enabling mobile phones as interfaces to cloud applications. Middleware 2009;2009:83–102.

Gordon MS, Jamshidi DA, Mahlke S, Mao ZM, Chen X. COMET: code offload by migrating execution transparently. In: Proceedings of the 10th USENIX conference on operating systems design and implementation (OSDI'12), Hollywood, California, USA, vol. 12; 2012. p. 93–106.

Goyal S, Carter J. A lightweight secure cyber foraging infrastructure for resource-constrained devices. In: Sixth IEEE workshop on mobile computing systems and applications, Lake District National Park, UK. IEEE; 2004. p. 186–95.

Gu X, Nahrstedt K, Messer A, Greenberg I, Milojicic D. Adaptive offloading for pervasive computing. IEEE Pervasive Comput 2004;3(3):66–73.

Huerta-Canepa G, Lee D. A virtual cloud computing provider for mobile devices. In: Proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond. San Francisco, USA: ACM; 2010.

Hung S, Shih C, Shieh J, Lee C, Huang Y. Executing mobile applications on the cloud: framework and issues. Comput Math Appl 2012;63(2):573–87.

Kant K. Data center evolution: a tutorial on state of the art, issues, and challenges. Comput Netw 2009;53(17):2939–65.

Kemp R, Palmer N, Kielmann T, Bal H. Cuckoo: a computation offloading framework for smartphones. In: Mobile computing, applications, and services. Seattle, WA, USA: Springer; 2012. p. 59–79.

Khan AN, MatKiah M, Khan SU, Madani SA. Towards secure mobile cloud computing: a survey. Future Gener Comput Syst 2013;29(5):1278–99.

Khan A, Othman M, Madani S, Khan S. A survey of mobile cloud computing application models. IEEE Commun Surv Tutorials 2014;16(1):393–413.

Kosta S, Aucinas A, Hui P, Mortier R, Zhang X. Thinkair: dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: IEEE proceedings INFOCOM. Orlando, FL, USA: IEEE; 2012. p. 945–53.

Koukoumidis E, Lymberopoulos D, Strauss K, Liu J, Burger D. Pocket cloudlets. ACM SIGPLAN Not 2012;47(4):171–84.

Kovachev D, Cao Y, Klamma R. Augmenting pervasive environments with an XMPP-based mobile cloud middleware. Mob Comput Appl Serv 2012a:361–72.

Kovachev D, Yu T, Klamma R. Adaptive computation offloading from mobile devices into the cloud. In: Proceedings of the 10th IEEE international symposium on parallel and distributed processing with applications. Madrid, Spain: IEEE; 2012b. p. 784–91.

Kumar K, Liu J, Lu Y-H, Bhargava B. A survey of computation offloading for mobile systems. Mob Netw Appl 2013;18(1):129–40.

Lee B. A framework for seamless execution of mobile applications in the cloud. In: Recent advances in computer science and information engineering; 2012. p. 145–53.

Lenders V, Wagner J, May M. Analyzing the impact of mobility in ad hoc networks. In: Proceedings of the 2nd international workshop on multi-hop ad hoc networks: from theory to reality. Florence, Italy: ACM; 2006. p. 39–46.

Marinelli E. Hyrax: cloud computing on mobile devices using mapreduce. Technical report, DTIC Document; 2009.

Mazzoleni P, Tai S. Engineering mobile field worker applications. In: International workshop on engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting. Cavat, Croatia: ACM; 2007. p. 75–9.

Motiwalla L. Mobile learning: a framework and evaluation. Comput Educ 2007;49(3):581–96.

Nazir F, Ma J, Seneviratne A. Time critical content delivery using predictable patterns in mobile social networks. In: International conference on computational science and engineering, 2009. CSE09'. Vancouver, Canada, vol.4. IEEE; 2009. p. 1066–73.

Ning W, Yang Y, Kun M, Yu C, Hao D. A task scheduling algorithm based on qos and complexity-aware optimization in cloud computing. In: National doctoral academic forum on information and communications technology; 2013. p. 1–8. http://dx.doi.org/10.1049/ic.2013.0202.

Oppermann R, Specht M. Adaptive mobile museum guide for information and learning on demand. In: Proceedings of 8th HCI international. Citeseer; 1999. p. 642–6.

Othman M, Hailes S. Power conservation strategy for mobile computers using load sharing. ACM SIGMOBILE Mob Comput Commun Rev 1998;2(1):44–51.

Ou S, Yang K, Liotta A. An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems. In: Proceedings of fourth annual IEEE international conference on pervasive computing and communications (PerCom06'). Pisa, Italy. IEEE; 2006a.

Ou S, Yang K, Zhang Q. An efficient runtime offloading approach for pervasive services. In: IEEE wireless communications and networking conference, 2006. WCNC 2006, Las Vegas, NV, USA, vol.4. IEEE; 2006b. p. 2229–34.

Patouni E, Kypriadis D, Alonistioti N. A lightweight framework for prediction-based resource management in future wireless networks. EURASIP J Wirel Commun Netw 2012;2012(1):1–12.

Qing W, Zheng H, Ming W, Haifeng L. CACTSE: cloudlet aided cooperative terminals service environment for mobile proximity content delivery. China Commun 2013;10(6):47–59. http://dx.doi.org/10.1109/CC.2013.6549258.

Rahimi MR, Ren J, Liu CH, Vasilakos AV, Venkatasubramanian N. Mobile cloud computing: a survey, state of art and future directions. Mob Netw Appl 2014;19(2):133–43.

Ramaboli A, Falowo O, Chan A. Bandwidth aggregation in heterogeneous wireless networks: a survey of current approaches and issues. J Netw Comput Appl 2012;35(6):1674–90.

Satyanarayanan M, Bahl P, Caceres R, Davies N. The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput 2009;8(4):14–23.

Shamshirband S, Daghighi B, Anuar NB, Kiah MLM, Patel A, Abraham A. Co-FQL: anomaly detection using cooperative fuzzy Q-learning in network. J Intell Fuzzy Syst 2015; in press.

Sookhak M, Talebian H, Ahmed E, Gani A, Khan MK. A review on remote data auditing in single cloud server: taxonomy and open issues. J Netw Comput Appl 2014;43:121–41.

Soyata T, Muraleedharan R, Funai C, Kwon M, Heinzelman W. Cloud-vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In: The eighteenth IEEE symposium on computers and communications (ISCC'17), Split Croatia; 2012. p. 59–66. http://dx.doi.org/10.1109/ISCC.2012.6249269.

Verbelen T, Stevens T, Simoens P, DeTurck F, Dhoedt B. Dynamic deployment and quality adaptation for mobile augmented reality applications. J Syst Softw 2011;84(11):1871–82.

Verbelen T, Simoens P, DeTurck F, Dhoedt B. AIOLOS: middleware for improving mobile application performance through cyber foraging. J Syst Softw 2012a;85(11):2629–39.

Verbelen T, Simoens P, De Turck F, Dhoedt B. Cloudlets: bringing the cloud to the mobile user. In: Proceedings of the third ACM workshop on mobile cloud computing and services. Ambleside, United Kingdom: ACM; 2012b. p. 29–36.

Wu S, Chao H, Ko C, Mo S, Jiang C, Li T, et al. A cloud model and concept prototype for cognitive radio networks. IEEE Wirel Commun 2012;19(4):49–58.

Yang K, Ou S, Chen H. On effective offloading services for resource-constrained mobile devices running heavier mobile internet applications. IEEE Commun Mag 2008;46(1):56–63.

Yang L, Cao J, Tang S, Li T, Chan A. A framework for partitioning and execution of data stream applications in mobile cloud computing. In: Proceedings of IEEE 5th international conference on cloud computing (CLOUD12'). Honolulu, Hawaii, USA: IEEE; 2012. p. 794–802.

Zekri M, Jouaber B, Zeghlache D. A review on mobility management and vertical handover solutions over heterogeneous wireless networks. Comput Commun 2012;35(17):2055–68.

Zhang J, Figueiredo R. Application classification through monitoring and learning of resource consumption patterns. In: 20th international parallel and distributed processing symposium (IPDPS06'). Rhodes Island, Greece: IEEE; 2006.

Zhang X, Jeong S, Kunjithapatham A, Gibbs S. Towards an elastic application model for augmenting computing capabilities of mobile platforms. In: Mobile wireless middleware, operating systems, and applications; 2010. p. 161–74.

Zhao B, Xu Z, Chi C, Zhu S, Cao G. Mirroring smartphones for good: a feasibility study. In: Mobile and ubiquitous systems: computing, networking, and services; 2012. p. 26–38.

Zheng X. Optimization techniques in communication networks [Ph.D. thesis]. University of Florida; 2008.