A Survey of Mobile Cloud Computing Application Models

Atta ur Rehman Khan, Mazliza Othman, Sajjad Ahmad Madani, *IEEE Member*, and Samee Ullah Khan, *IEEE Senior Member*

***Abstract*—Smartphones are now capable of supporting a wide range of applications, many of which demand an ever increasing computational power. This poses a challenge because smartphones are resource-constrained devices with limited computation power, memory, storage, and energy. Fortunately, the cloud computing technology offers virtually unlimited dynamic resources for computation, storage, and service provision. Therefore, researchers envision extending cloud computing services to mobile devices to overcome the smartphones constraints. The challenge in doing so is that the traditional smartphone application models do not support the development of applications that can incorporate cloud computing features and requires specialized mobile cloud application models. This article presents mobile cloud architecture, offloading decision affecting entities, application models classification, the latest mobile cloud application models, their critical analysis and future research directions.**

***Index Terms*—Mobile Cloud Computing, Application Models, Mobile Cloud Application Models, Cloud Computing, Mobile Cloud Survey, Mobile Cloud Application Models Survey.**

I. Introduction

CLOUD computing is a coalesce of many computing fields and has gained much popularity in the recent years. Cloud computing provides computing, storage, services, and applications over the Internet. Moreover, cloud computing facilitates to reduce capital cost, decouple services from the underlying technology, and provides flexibility in terms of resource provisioning.

Similarly, smartphones are also gaining enormous popularity due to the support for a wide range of applications, such as games, image processing, video processing, e-commerce, and online social network services. As smartphone applications increase in complexity, so do their demand on computing resources. Unfortunately, the advances in smartphone hardware and battery life have been slow to respond to the computational demands of applications evolved over the years. Therefore, many applications are still unsuitable for smartphones due to constraints, such as low processing power, limited memory, unpredictable network connectivity, and limited battery life [1], [2].

In general, to make the smartphones energy efficient and computationally capable, major hardware and software level changes are needed, which requires the developers and manufacturers to work together [3], [4], [5]. Due to size-constraints, hardware level changes alone may not enable smartphones to achieve true unlimited computational power. Therefore, software-level changes are more effective, where computation is performed on remote resources with partial support of a smartphone's hardware [6].

Computation offloading is a procedure that migrates resource-intensive computations from a mobile device to the resource-rich cloud, or server (called nearby infrastructure). Cloud based computation offloading enhances the applications performance, reduces battery power consumption, and execute applications that are unable to execute due to insufficient smartphone resources. Moreover, cloud offers storage services [7] that can be used to overcome the storage constraints of the smartphones. Currently, many applications exist with cloud support for multiple domains, such as commerce [8], healthcare [9], [10], education [11], [12], social networks [13], gaming [14], file sharing [15], and searching [16], among others.

We define *mobile cloud computing as an integration of cloud computing technology with mobile devices to make the mobile devices resource-full in terms of computational power, memory, storage, energy, and context awareness.* Mobile cloud computing is the outcome of interdisciplinary approaches comprising mobile computing and cloud computing. Therefore, this transdisciplinary domain is also referred as *mobicloud computing* [17].

The term *mobile cloud* is generally referred to in two perspectives: (a) infrastructure based, and (b) ad-hoc mobile cloud. In infrastructure based mobile cloud, the hardware infrastructure remains static and provides services to the mobile users. Alternatively, ad-hoc mobile cloud refers to a group of mobile devices that acts as a cloud and provides access to local or Internet based cloud services to other mobile devices. In this survey, we limit the selection of application models to the former case namely, the infrastructure based mobile cloud. Therefore, ad-hoc mobile cloud based systems/application models [18] and associated issues, such as mobility of cloud infrastructure and geo-distribution of service nodes [19], [20], are beyond the scope of this survey.

Although cloud is useful for computing and storage [21], [22], [23], the traditional computation offloading techniques cannot be used for the smartphones directly, because these techniques are generally energy-unaware and bandwidth-hungry. Moreover, the traditional mobile application models support the development of applications that can execute only on mobile devices without computation offloading. However,

Manuscript received August 1, 2012; revised January 7, 2013 and March 21, 2013.

A. R. Khan and M. Othman are with the University of Malaya, Kuala Lumpur, Malaysia (e-mail: attaurrehman@siswa.um.edu.my).

S. A. Madani is with COMSATS Institute of Information Technology, Abbottabad.

S. U. Khan is with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, USA.

Digital Object Identifier 10.1109/SURV.2013.062613.00160

TABLE I
CLOUD AND MOBILE CLOUD COMPUTING COMPARISON

Issues	Cloud Computing	Mobile Cloud Computing
Device energy	×	√
Bandwidth utilization cost	×	√
Network connectivity	×	√
Mobility	×	√
Context awareness	×	√
Location awareness	×	√
Bandwidth	×	√
Security	√	√

there are a few applications that utilize cloud resources, but the usage is limited to only storage and application-specific services, such as Apple's Siri (voice based personal assistant) and iCloud storage service. Therefore, smartphones require an application model that supports computation offloading and is optimized for mobile cloud environment in terms of heterogeneity, context awareness, application partitioning overhead, network data cost, bandwidth, and energy consumption.

There are various generic surveys that highlight the importance of mobile cloud computing. In [24], the authors discuss two mobile cloud application models (Hyrax [25], cloudlets [2]) and emphasize on the importance of intelligent access schemes [26]. In [27] and [28], the authors discuss generic issues of a mobile cloud. In [29], the authors present basic level comparison of the application models. However, most of the techniques discussed in [29] are not applicable to infrastructure based cloud environments or cannot be classified as application models. Also, the authors did not identify the issues associated with the application models.

The main contribution of this article is to survey the most recent mobile cloud application models (work done between 2008-2012) and highlight their strengths, weaknesses and issues that need further attention. We also present the differences between cloud and mobile cloud computing, mobile cloud architecture, and the main entities that impact the overall computation offloading decision. Moreover, we highlight the parameters that affect mobile cloud application models, and present classification of application models. Furthermore, we compare and critically analyze the application models along their critical outstanding issues, and suggest future research directions.

The rest of the paper is structured as follows. Section II presents mobile cloud architecture, computation offloading workflow, and entities that affect computation offloading process. Section III presents the criteria for comparison of mobile cloud application models. Section IV discusses the mobile cloud application models and highlights their advantages and shortcoming. Section V presents comparison of application models, and mobile cloud applications. Finally, Section VI concludes with a discussion of critical outstanding issues and future research directions in this area.

II. Mobile Cloud Architecture & Computation Offloading

The main objective of cloud computing is to facilitate small businesses in a cost effective fashion to provide access to technologies that are beyond their reach. By using cloud computing, small businesses can expand their IT resources based on service demands and avail equal opportunities of growth to compete with other businesses within the market. Alternatively, the primary objective of mobile cloud computing is to provide enhanced user experiences to mobile users that may be in terms of computation time, battery life, communication, services, and mobile device resource enhancement. Therefore, both of these technologies have different objectives and challenges. For instance, in mobile cloud computing, the network connectivity, amount of communication, bandwidth utilization cost, and mobile device energy are considered to be the foremost issues, which may not be the case (or least important) in cloud computing. However, the mobile cloud application models are based on the standard cloud service model that includes Infrastructure as a Service (IaaS) [30], Platform as a Service (PaaS) [31], and Software as a Service (SaaS) [32], [33]. Therefore, based on the working of the application models, any of these service layers can be utilized. Some of the well-known services for mobile cloud computing include Amazon Elastic Compute Cloud (EC2) [34], Google App Engine [31], and Microsoft Azure [35].

To the best of our knowledge there is no standard definition or feature characterization of mobile cloud computing. Therefore, we present the possible best comparison of cloud and mobile cloud computing in terms of significance of the issues listed in Table I.

A. Mobile Cloud Architecture

In the current mobile cloud architecture, mobile devices can access cloud services in two ways, i.e., through mobile network (telecom network) or through access points, as shown in Figure 1.

In the mobile network (telecom network provider) case, the mobile devices such as cellular/satellite smartphones [36] are connected to a mobile network through a Base Station (BS) or via a satellite link. However, if the smartphones are not equipped with a satellite communication module, then external satellite communication devices [37] are used. The telecom networks are further connected to the Internet and provide Internet connectivity to the users. Therefore, if the users have mobile network connectivity, the users can access cloud based services through the Internet.

In the access point case, the mobile users connect to the access points through Wi-Fi that is further connected to the Internet service provider to provide Internet connectivity to the users. Therefore, the mobile cloud users can access cloud based services without utilizing telecom services, which may

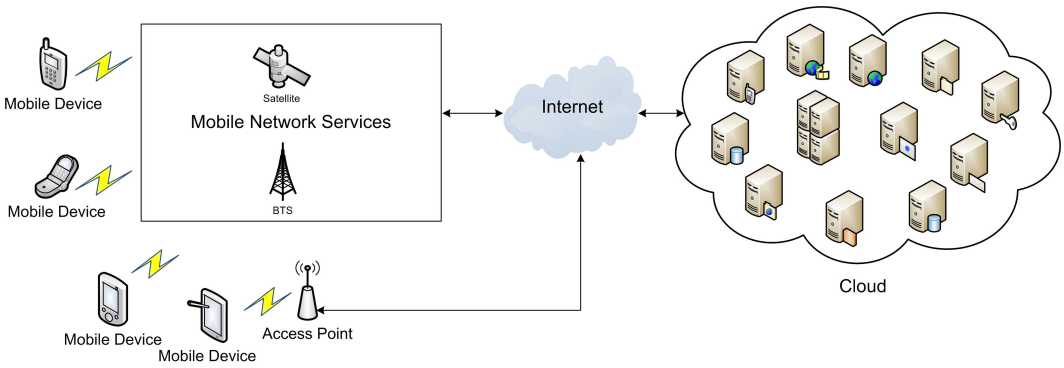

Fig. 1. Mobile cloud architecture

charge them for data traffic. Moreover, Wi-Fi based connections provide low latency and consume less energy compared to 3G connections [14]. Consequently, mobile cloud users prefer to use Wi-Fi Internet connections whenever accessible.

B. Computation Offloading Decision Making

A mobile cloud application goes through the following steps before offloading computations to the cloud. Figure 2 presents the basic workflow of the computation offloading process. The workflow starts with the execution of an application followed by checking the user's offloading permission. If offloading is enabled, then application checks connectivity to the cloud resources and notes the available/assigned resources. The next step involves deciding whether offloading is favorable, depending on the users' desired objective (discussed in Section IV). If it is favorable, then the computation offloading is performed. Otherwise, the application performs all computations locally.

The decision of computation offloading is an extremely complex process and is affected by different entities, for instance user, connection, smartphone, application model, application (nature) and cloud service. Figure 3 presents different entities that can affect the computation offloading decision in multiple ways.

1) User: A user may enable or disable the computation offloading based on network data cost, cloud service cost, importance of data privacy and job turnaround time. Moreover, the decision is also dependent on the users' desired objective. For instance, a user may be interested in saving energy, enhancing application performance or executing an application that does not have sufficient resources on the smartphone.

2) Connection: Different communication technologies have their own limitations. For instance, Wi-Fi based connections provide high bandwidth and shorter delays. Alternatively, 3G connections provide lower bandwidth and suffer from higher delays compared to Wi-Fi connections [14]. Therefore, if both connections are available, then user may prefer to use Wi-Fi connection. However, Wi-Fi connectivity is not always feasible, particularly in mobile environments. Therefore, 3G/4G connections that charge for bandwidth usage are used. Hence, from a connection point of view, the computation offloading decision can be affected by network bandwidth, delay, and cost.

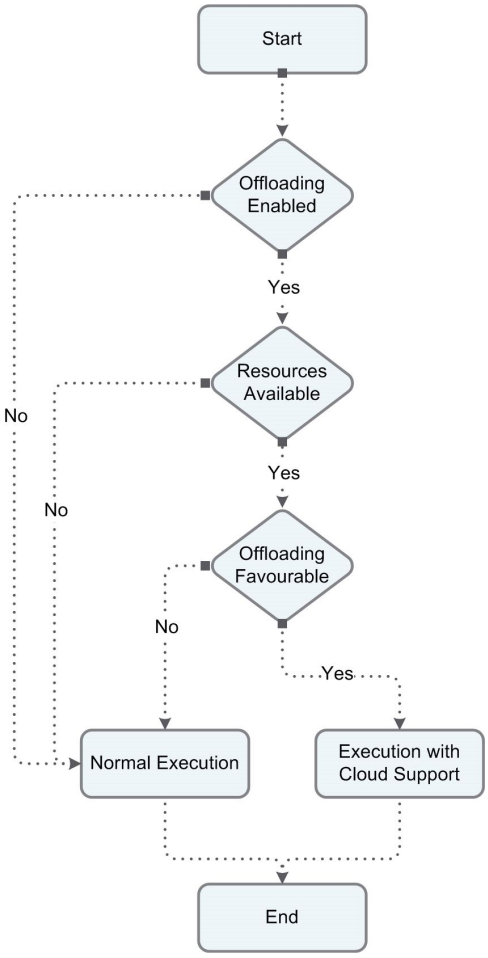

Fig. 2. Process of computation offloading

3) Smartphone: The smartphones have achieved great development in terms of hardware resources in the past few years. The latest smartphones are equipped with high performance processors, memory, sensors and storage. For instance, Sony Xperia S [38] comes up with 1.5GHz Dual Core processor, 1GB RAM, 32GB data storage support, and 1750mAh battery. Similarly, HTC One X [39] has 1.5Ghz Quad-core processor, 1GB RAM, 32GB data storage support, and 1800mAh battery. Therefore, it is obvious that users with high performance smartphones may require mobile cloud support less frequently, compared to the users that have low

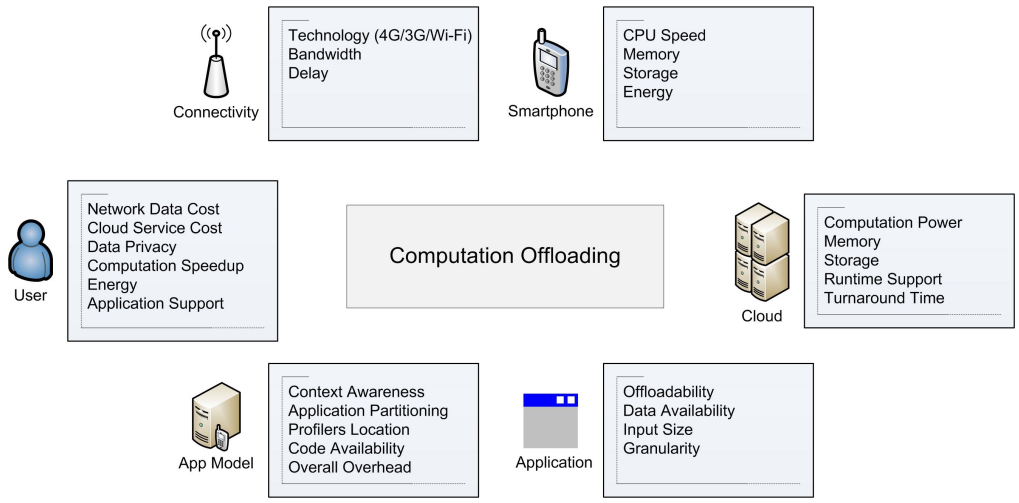

Fig. 3. Entities affecting computation offloading

performance smartphones and runs out of resources quickly.

4) Application Model: The mobile cloud application models differ from each other in terms of design and objectives. For instance, the objective of computation offloading may be energy efficiency, application performance or application execution for devices that do not have sufficient resources. The application models may also differ in terms of context awareness, application partitioning, code availability in the cloud, profilers and overhead. A detailed discussion of these parameters is provided in Section III.

5) Application: The computation offloading decision also depends on the nature of the application. For instance, an application that requires local hardware resources (GPS, camera, and sensors) may not be able to execute in the cloud unless the application is partitioned into components, and local-resource independent components are moved to the cloud. Similarly, if the application data is unavailable in the cloud and the input data size is too large, then smartphone side computation may be favorable. Alternatively, transferring a large amount of data may incur higher turnaround time and consume higher energy in terms of communication, which may offset the benefits of offloading.

6) Cloud Service: The selection of cloud service is very crucial for computation offloading. Therefore, if a user requires mobile cloud support for computation offloading, then it is important that the cloud must have runtime support for the offloaded application/component. Moreover, the leased cloud service must be rich in resources in order to gain advantage of computation offloading. For example, if a smartphone and a Virtual Machine (VM) [40], [41] (deployed in the cloud) have the same specifications (computational power, memory), then the user may not get any improvement in the application performance. Although the scenario may be beneficial in terms of energy (depending on data/code size), it is not beneficial for enhancing application performance. In fact, the application performance may decrease due to the additional computation and delay involved in the offloading process.

All of the above mentioned entities have vital importance in the decision of computation offloading. However, it is the responsibility of the application models to perform the computation offloading considering the related entities. Therefore, in this survey we mainly focus application models taking into account the related entities.

III. Criteria for Comparison of Application Models

This section highlights various parameters that play a vital role in the acceptance of any mobile cloud application model. Therefore, the mobile cloud application models that address most of the following parameters are considered to be preeminent. The selected mobile cloud application models presented in section IV are compared based on the parameters discussed in following subsections.

A. Context Awareness

Context awareness of an application model refers to its awareness about the entities and parameters that can affect the decision of computation offloading. In principal, it is very important for an application model to be context aware because static offloading is not always beneficial, and cases may occur where the performance of the applications degrade with the computation offloading [42]. The computation offloading decision affecting entities and parameters are already discussed in Section III.

B. Latency

In mobile cloud computing, latency is defined as the time involved in offloading the computation and getting back the results from the nearby infrastructure or cloud, sometimes referred as turnaround time. The latency depends on multiple factors, such as offloaded code size, data input size, location of the required data, offloading scheme and granularity, network bandwidth, execution delay, and resultant data size.

In practice, the latency of an application model may vary from case to case. For instance, in clone based application models, if a smartphone and its clone are synchronized on short time intervals, then synchronization may not be required before offloading and the latency may be low. Alternatively, if the synchronization is done after long time intervals or on

demand, then synchronization will be required before computation offloading so that the execution is done accurately. However, the runtime synchronization may increase latency.

Moreover, some application models require installations on the nearby infrastructure, and the installation time may vary from a few seconds to a few minutes[2], [43]. Also, the computation platforms are not always in a ready state and may require seconds to minutes to startup the service [44]. Therefore, the service startup delay makes the application models unsuitable for real-time applications.

C. Bandwidth Utilization

In the application models, bandwidth utilization refers to the amount of data/code migrated to offload the computation. Therefore, if the computation offloading requires a large amount of data to be transferred on runtime, then higher latencies may occur. Alternatively, if the data is offloaded to the cloud in advance to reduce the offloading latency, then synchronization of the data is required as discussed in the previous section. However, if the synchronization is done on short time intervals, then the communication may lead to high bandwidth usage that is not free in cellular networks. Nevertheless, costly bandwidth is not the only primary concern. Bandwidth in the wireless networks is limited as compared to the wired networks, particularly in cellular networks. The 4G technology [45] aids to narrow down the bandwidth gap between wireless and the wired networks. However, there still exist many issues related to the access protocols and the network architecture. Even if the current 4G issues are ignored, the cellular network bandwidth utilization may never be free that makes efficient bandwidth utilization an important concern. Therefore, during the development of the mobile cloud application models, efforts must be made to optimize and negotiate a tradeoff between bandwidth utilization and latency.

D. Generality

The generality of an application model refers to its support for a range of applications. In practice, there are multiple types of applications with different resource demands and behavior. For instance, tasks like scanning files for viruses, indexing files for quick search, and crawling news website for the latest news are delay tolerant tasks that do not require user interaction after initiation. Once the tasks are completed, the results can be synced back to the smartphones. Alternatively, the applications used for the image, speech, and video processing may require quick response from the cloud to smoothen the interaction between the user and the smartphone application. Consequently, it is quite challenging for an application model to support multiple types of applications. However, efforts must be made to design mobile cloud application models that can support all types of applications.

E. Privacy

With the advancements in the mobile device technology, sensors such as GPS have become cheap, and are available in nearly all the latest smartphones. Many recent applications [46], [47], [48] require user location to deliver location-based services. These services are either user-invoked to get location related information [49], or service-provider invoked to deliver location-based ads. For instance, many free mobile applications [50], [51], require GPS access to show location-based ads against the free services availed by the users. Therefore, the location information of the users can cause serious privacy issues, particularly when other user-related information is already known [52], [53].

Similarly, data privacy is also important and is one of the main bottlenecks that restrict consumers from adopting mobile cloud computing. The users' data stored in the cloud may include emails, tax reports, personal images, salary and health reports etc, and may contain sensitive information. Therefore, the consumers cannot afford any privacy leakage as it may lead to financial loss and legal issues [54]. The European Union has passed some laws [55], [56] for the handling of data, according to which the data storage servers must reside in the countries that can provide sufficient protection. Moreover, in some cases the data storage location must be known. However, this is not always possible in a cloud environment due to the absence of standards, data privacy, and cloud security [57]. Therefore, to gain consumers trust in the mobile cloud, the application models must support application development with privacy protection and implicit authentication mechanisms [58], [59].

F. Complexity

The applications developed for the mobile cloud platforms must be able to execute in both online and offline mode. Moreover, the applications must utilize minimum bandwidth with considerable delay. Therefore, some models (discussed in Section IV) partition the applications into manageable and off-loadable components (sub-partitions) that can move to the cloud with minimal bandwidth requirement.

The application component offloading can be done in two ways, i.e., static and dynamic. In static offloading, the programmers pre-determine the application components that can be offloaded to the cloud. However, this solution is not very effective, as many entities may affect the computation offloading (discussed in Section III). Alternatively, in dynamic (also called context aware) offloading, the execution location of the components is not pre-determined, and the offloading decisions are made intelligently by analyzing contextual information, such as, smartphone resources, bandwidth, latency, energy, and cloud resources.

Moreover, some application models use parallelism in the cloud to reduce execution delay, but depending on the cloud service, the parallelism may not always be supported [60]. In [61], [62], [63], the authors propose multiple techniques to achieve Quality of Service (QoS) by reducing latency. The features, such as application partitioning, dynamic offloading, resource monitoring, contextual information analysis, and cloud parallelism add to the complexity of the application models. Therefore, complex mobile cloud application models may be difficult to implement and may incur high overhead on the smartphones in terms of computation, memory and energy.

G. Security

Security is one of the most prominent bottlenecks in the adoption of cloud computing [64]. Cloud computing endure a number of security issues, for instance, data access control [65], data distribution over a distributed infrastructure, data integrity, service availability, and secure communication. Also, the mobility adds some additional security issues [66], [67] that make mobile cloud security more challenging. Another security issue that requires concern is the provisioning of virtually unlimited resources to untrustworthy users [68]. For example, an adversary may use virtually unlimited resources against the enterprises, and cause problems for the victims and the cloud service providers.

In mobile cloud computing, security needs to be analyzed from two perspectives, i.e., the smartphone and the cloud. The smartphones must be clean from the malicious codes, such as viruses, trojan horses, and worms. The malicious codes are security threats and can change an application's behavior, which may cause privacy leakage or data corruption. Therefore, to keep the smartphones clean from the malicious codes, security applications [69], [70], [71], must be used regularly. However, the scanning process of the security applications is a computation-intensive task that consumes high energy. Therefore, it is not feasible for the smartphones to execute security applications for extended periods. In [72], [73], [74], the authors propose multiple techniques that perform computation offloading of (malicious code scanning) resource-intensive tasks to achieve security and gain energy efficiency. Alternatively, from the cloud security perspective, the data stored in the cloud can be lost, altered, denied, or leaked. Therefore, the data stored in the cloud must have multiple backups with integrity support to avoid data loss and undesired modifications. In [74], [75], [76], [77], the authors propose multiple techniques that focus data integrity issues. Nevertheless, security is one of the important issues of the mobile cloud computing and demands serious consideration during the development and adoption of application models.

H. Programming Abstraction

The cloud platforms support different APIs, data models, query languages, and cost models. Similarly, the smartphones run different operating systems that have variable hardware and software requirements. Therefore, the heterogeneities in smartphones and cloud platforms make the development of mobile cloud applications complicated. However, the heterogeneities arise due to the lack of standards and sometimes self created by the vendors to retain the users.

To facilitate the programmers in development of mobile cloud applications, new tools are required that provide programming abstraction and hide the underlying complexities of the cloud and smartphones. Moreover, the new tools must enhance the performance of applications, and allow programmers to control the behavior and execution location of the applications [78], [79], [80]. For instance, MapReduce [81] and Hadoop [82] allow the development of applications without knowing the underlying operational complexity, and make the coding easier for the programmers. Therefore, during the development of mobile cloud application models, the heterogeneity issues must be considered, and effort must be made to keep the programming abstraction high, so that the developers can easily adopt new programming tools and application models.

I. Scalability

Scalability is one of the most important features of cloud computing. Therefore, the mobile cloud application models must support the development of applications that can scale in the cloud to meet unpredictable user demands. Moreover, the application models must enhance the supported features to incorporate new types of applications in a timely manner. Nevertheless, the mobile cloud application models must also be scalable in terms of adoption. For instance, an application model that requires nearby computational infrastructure and demands heavy software installations is less scalable compared to the application model that is based on the cloud platform having no hardware setup requirement. However, the scalability is not only dependent on the application model, and to some extent depends on the cloud platform. For instance, in Amazon EC2 [34] users can control almost the entire software stack. This feature restricts Amazon's ability to provide automatic scalability as the replication control becomes highly application dependent. Alternatively, Google AppEngine [31] focuses on the traditional web applications with stateless computation and stateful data storage that makes the applications impressively scalable. Therefore, the aforementioned scalability issues must be considered during the development or adoption of the mobile cloud application models.

J. Execution resource

The mobile cloud applications execute in two ways [2], [83]. In the first case, the applications execute on the nearby infrastructure that acts as a (virtual) cloud, for instance, personal computers, laptops, and servers. In the second case, the applications execute in a real cloud, for instance, Amazon EC2 [34], Google App Engine [31], and Microsoft Live Mesh [84]. Therefore, the mobile cloud application models may support execution of the applications on either nearby infrastructure, cloud or both.

Execution resource significantly affects the scalability and availability of the application models. For instance, the availability of a nearby infrastructure is an unrealistic assumption, particularly when the user is on the move. Therefore, the assumption may be valid only for home and office environments, where the personal computers or nearby servers are available. However, some application models may require heavy software installations on the infrastructure to support computational offloading. In principal, the personal computers do not promise virtually unlimited resources like real cloud platforms. Moreover, keeping the personal computers always in the ready state, just for the sake of computation offloading is not an energy efficient solution. Therefore, to make the application models scalable and capable of utilizing virtually unlimited resources with guaranteed availability; shifting the task of computation from the nearby infrastructure to the real cloud platforms is an appealing choice. Nevertheless, cloud

computing is more energy efficient and the researchers have proposed different energy efficient techniques [85], [86], [87], [88] that can get maximum output from the cloud based servers.

K. Platform

A platform is the underlying software technology of the smartphones on which the application models are based. Smartphones manufactured by different manufacturers can be grouped together based on the operating systems that run on the devices. The renowned smartphone operating systems are Android [89], iOS [90], Symbian [91], Mobile operating system [92] and BlackBerry OS [93].

- *Android* is an open source operating system powered by Google, and its kernel is based on Linux. Android OS supports Java based application.
- *iOS* is a proprietary OS of Apple and is based on MAC OS X. iOS applications are mainly developed in objective C.
- *Symbian* is an open source OS powered by Nokia, while its applications are developed in Java and C++.
- *Mobile OS* is a proprietary of Microsoft and support applications developed on .Net framework. Nokia has also announced that its newly manufactured smartphones will be running Windows Phone 7 powered by Microsoft [94].
- *BlackBerry OS* is a proprietary of Research in Motion (RIM) and its applications are mainly developed in Java.

Most of the application models discussed in Section IV supports a single platform due to the heterogeneity of the underlying technologies, and the variety of supported programming languages. For example, Apple iOS does not support Java-based applications, and its applications are purely coded in Objective C. Moreover, some mobile operating systems are not designed for computational offloading, for instance, the Google Android application model has more support for computational offloading compared to Apple iOS.

IV. Application Models for Mobile Cloud Computing

The mobile cloud application models are designed to achieve a particular objective, such as executing applications that have insufficient resources for local execution, enhancing applications performance (in terms of computation time), or achieving energy efficiency on mobile devices. In some scenarios, a single application model may achieve multiple objectives. On the contrary, achieving one objective may affect others. For example, if the primary objective of an application model is to achieve energy efficiency, then certain cases may occur in which performance is sacrificed. Therefore, the application models must be adopted considering the objective(s) and their affect on the counterparts. From the design perspective, the models that support multiple objectives are considered to be preeminent due to support for wide range of applications and scenarios. Based on the objective(s) of the surveyed application models, we classify the mobile cloud application models into four categories that are listed as follows.

A. Performance Based Application Models

The primary objective of performance based application models is to enhance the performance of mobile device applications by utilizing cloud resources. Therefore, the resource intensive computations are offloaded to the high speed cloud where the computation is performed in less time compared to the mobile device. Consequently, applications execute on mobile devices with enhanced performance (in terms of computation time) by utilizing cloud resources.

1) CloneCloud: CloneCloud [95] is based on augmented execution technique that offloads parts of application execution to the nearby infrastructure or cloud. CloneCloud does not require programmer support for the conversion of applications (for cloud environment), and offloads parts of the unmodified application execution from the mobile device to the smartphone clone (in the cloud). The synchronization of the smartphone and its clone is very important for consistent execution. Therefore, when augmentation is required, the smartphone application process enters a sleep state and transfers the process state to the clone. The VM creates a new process state and overlays the received information, followed by execution of the clone. On completion of the execution, the process state of the clones' application is sent to the smartphone, where the process state is reintegrated into the smartphones' application and the application comes out of a sleep state.

CloneCloud supports five types of augmented executions [83]: (a) *primary functionality outsourcing* that offloads all resource intensive applications to the cloud, whereas the user interface and light weight processing are left on the smartphone, (b) *background augmentation* that handles system functionalities that do not require frequent user interactions are moved to the cloud while the results are synced with the smartphone on completion, (c) *mainline augmentation* that facilitates in debugging application issues, such as data leakage, fault tolerance and memory leakage, (d) *hardware augmentation* that deals with the performance enhancement of the clone by tweaking VM settings, and (e) *augmented throughput multiplicity* that deals with the parallel execution of the clones and scheduling decisions to gain application performance.

In CloneCloud, the process of application partitioning is fully dynamic. Therefore, an application is analyzed and a static flow control graph is generated that makes a partitioned graph and facilitates in the application partitioning. When an application executes, the threads migrate from the mobile device to the cloud at auto selected points. However, the selection of migrate-able points from where threads can migrate to the cloud is a challenging task as it can affect the overall performance gain of the applications. An example of program partitioning and thread migration is shown in Figure 4.

The selection of migrate-able points is done with the help of a partitioning component that uses static analyzer, dynamic profiler and mathematical optimizer/optimization solver. The *static analyzer* is responsible for the analysis of potential migration points (partition choices) and associated constraints, such as mobile device hardware access, native state sharing and cyclic migration. The *dynamic profiler* is responsible for the collection of cost metrics data that facilitates in the

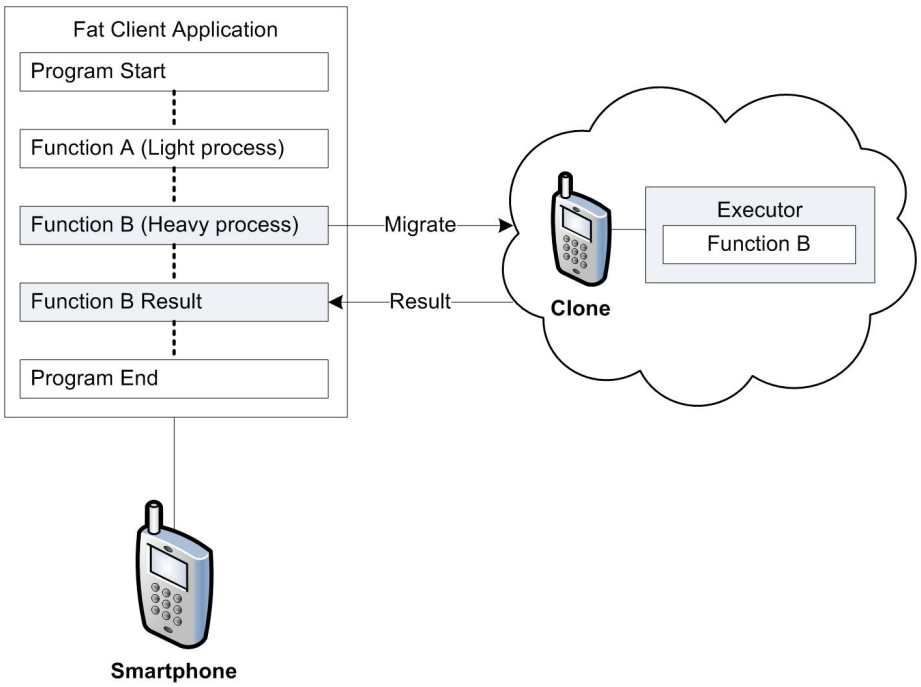

Fig. 4. Program partitioning and thread migration [95]

development of cost models, used for offloading decision making. The cost metrics data can be collected by running the executable repeatedly on both sides (smartphone, cloud) with different input settings. Lastly, *the mathematical optimizer/optimization solver* is responsible for optimum selection of migration points using cost models. Figure 5 shows high level architecture of the CloneCloud.

The main components of the CloneCloud are node manager, migrator, database, profiler (dynamic profiler) and partition analyzer. The *migrator* is responsible for suspending, packaging, resuming, and merging the thread states on both sides. The *node manager* performs provisioning, image synchronization, and handles communication between the migrated threads. Lastly, the *database* is responsible for record keeping of application partitions. Chun *et al.* tested CloneCloud prototype for three tasks, i.e., virus scan, image search and behavior profiling [95]. The results show that CloneCloud based applications gained 21.2% performance improvement in terms of execution time.

The advantage of this model is that when a smartphone is lost or destroyed, the clone can be used as a backup for the recovery of data and applications. Moreover, CloneCloud augments execution of the smartphone applications on the cloud by performing a code analysis for application partitioning, taking into consideration the offloading cost and constraints. CloneCloud also supports fine-grained thread-level migration that is more beneficial compared to the traditional suspend-migrate-resume mechanisms [96]. Considering the shortcomings, the model is only capable of migrating at points in the execution where no native heap state is collected. Moreover, CloneCloud requires the development of cost model for every application under different partitions, where each partition is executed separately on the mobile device and the cloud. Therefore, the execution of partitions on mobile device for the development of cost model may consume extra energy. Furthermore, to fit all of the proposed augmentation types, basic and fine-grained synchronization is required between the smartphone and the clone that may be resource intensive in terms of bandwidth utilization and energy consumption. Nevertheless, the authors assume that the cloud environment is secure that is not always the case. In CloneCloud, the privacy of data and piracy of applications is of high concern from the clones' perspective. For example, if an adversary gets a clone of the smartphone from the cloud, then the clone can be easily installed on the same model of the smartphone. Therefore, the adversary may use the clones' data and installed applications that may lead to data privacy and application piracy issues.

2) Zhang et al. Model: Zhang *et al.* [97] propose a model that is based on elastic applications technique, where a single elastic application is partitioned into multiple components called weblets. A *weblet* can be defined as an independent functional unit of an application that can compute, store, and communicate while keeping its execution location transparent. The offloading decision of the weblets depends on factors such as CPU load, memory, network conditions, user preferences and battery level. Moreover, the weblets can be platform-independent or platform-dependent, based on the programming technology used. The topology of the elastic applications falls into multiple types of patterns, called *elasticity patterns*. Weblets support three types of elasticity patterns, i.e., replication, splitter and aggregation as shown in Figure 6.

The *replication pattern* supports two types of replications. In the first type, multiple replicas of a weblet execute in the cloud to complete a single task. This type of replication is very useful in reducing execution time and latency, particularly for applications that can be divided into similar tasks, for instance, scanning files and processing a set of images. The second type of replication is useful for situations where

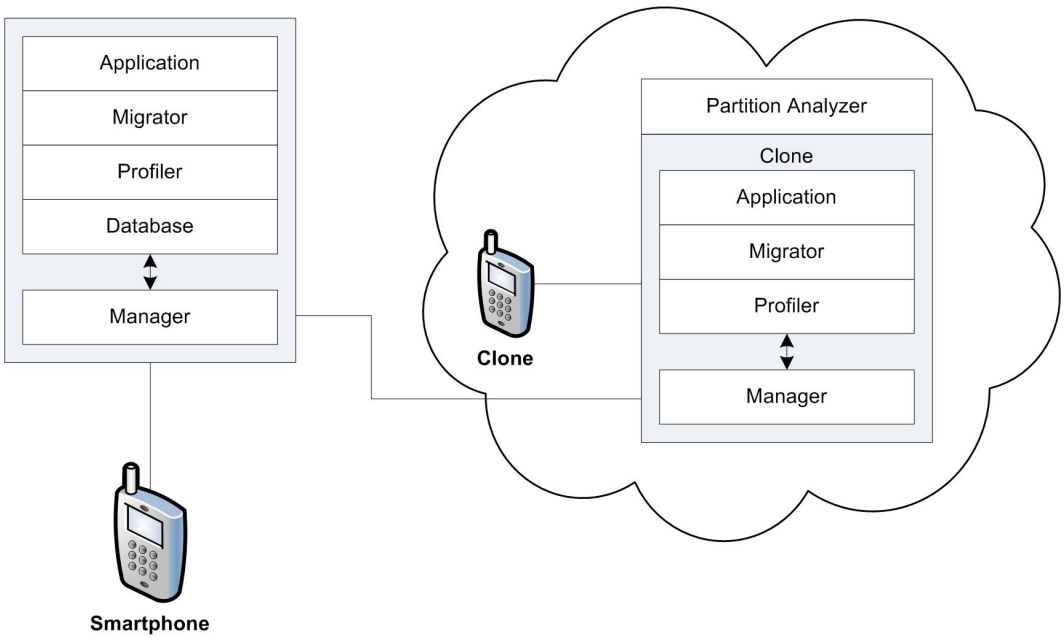

Fig. 5. CloneCloud architecture [95]

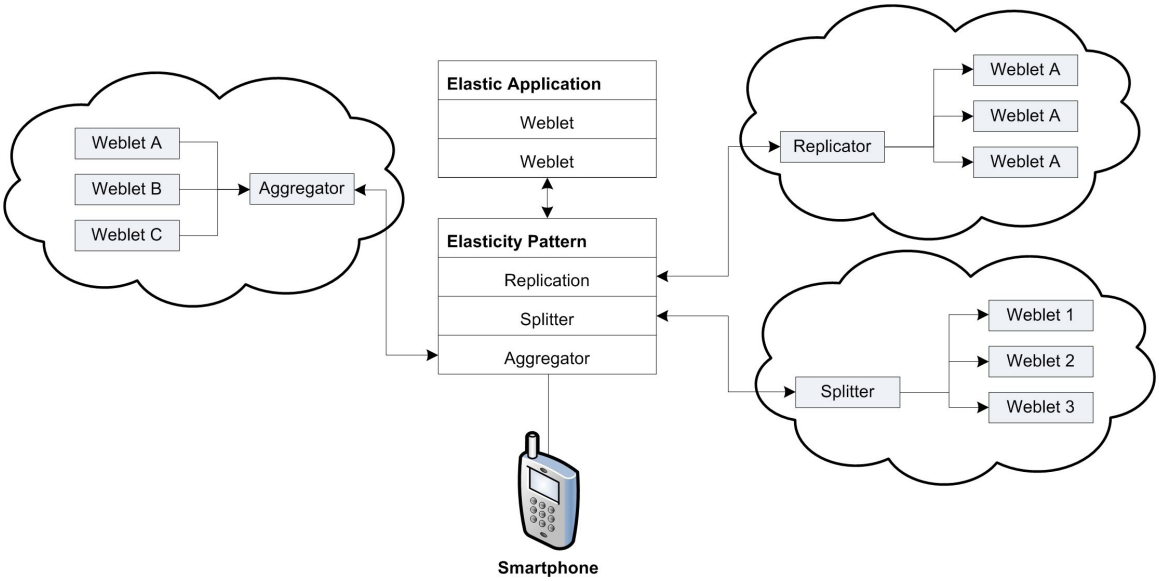

Fig. 6. Elasticity patterns [97]

applications cannot be divided into similar tasks. Therefore, multiple replicas of a weblet execute in the cloud, and the one that completes its execution first returns the result to the smartphone. Consequently, to some extent it reduces the latency and provides fault tolerance.

The *splitter pattern* executes different implementations of a weblet on a shared resource. The splitter pattern increases the extensibility of the applications by adding new implementations to a shared resource without changing the application structure. Moreover, splitter pattern is useful in enhancing user experience as it congregates multiple services on a single device. For instance, different weblet implementations can get user data from different social networks and provide a unified interface to the user for accessing multiple social network services.

The *aggregator pattern* runs multiple weblets in the cloud that monitor user web accounts and services, such as emails and instant messages. Therefore, whenever some account activity occurs, the weblets relay the aggregated information to the device using weblet push. In some cases, the splitter and aggregator patterns may work together, where splitter pushes the request to the weblet and the aggregator pushes the data to the device.

The elastic applications have three main components, i.e., user interface, weblet(s) and manifest. The *User Interface (UI)* is used for interaction with the applications. The *weblet* is an independent functional unit of an application and executes on the device or cloud. Lastly, the *manifest* is a static XML file that contains information about the application requirements and constraints, such as processing power, storage, network

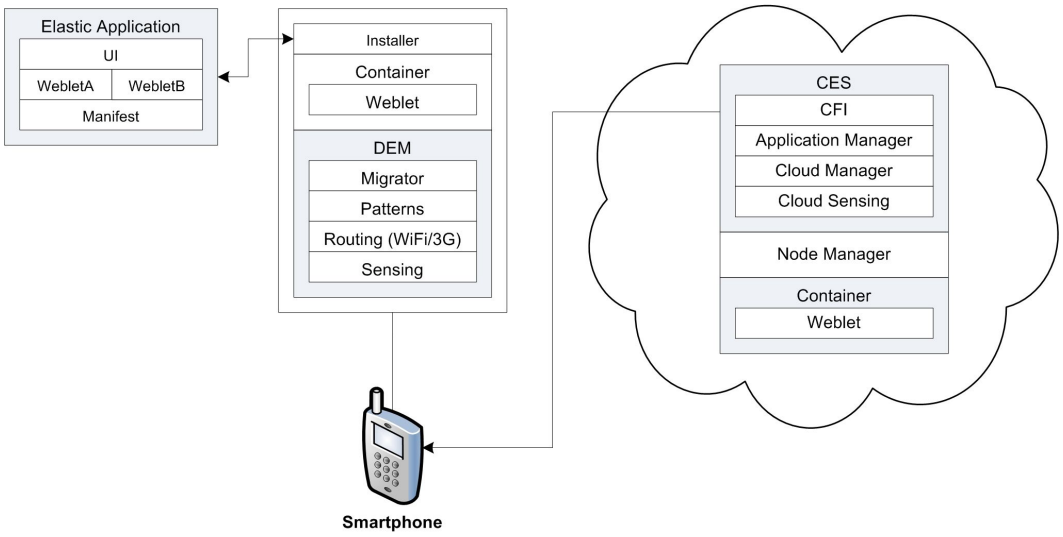

Fig. 7. Zhang *et al.* model architecture [97]

connection, execution time and execution location. The main architecture of Zhang *et al.* model is illustrated in Figure 7.

On the smartphone side, *Device Elasticity Manager (DEM)* is the main component that is responsible for application configuration (at launch time and on run-time) and weblet migration. The application configuration includes information about weblet replication, execution location, communication path (3G, WiFi), processor utilization, battery state, and the required hardware (sensors). Alternatively, *Cloud Elasticity Service (CES)* is one of the main components that resides on the cloud side and consists of four sub-components, (1) a *cloud manager* that is responsible for provisioning and monitoring of resources on the cloud that are provided to the weblets, (2) an *application manager* that launches and maintains the weblets on the cloud platform, (3) a *cloud sensing* that collects information about weblets resource consumption and provides that information to the cloud manager, and (4) a *Cloud Fabric Interface (CFI)*, which is a service provided to the elastic applications/smartphones. Moreover, CFI facilitates the migration of weblets between the smartphones and the cloud. Also, the cloud side contains a *node manager* that is responsible for monitoring the overall cloud node (server) resources.

Among the advantages of Zhang *et al.* model is a wide range of elasticity patterns to optimize the execution of applications according to the users' desired objectives. Consequently, the offloading decisions of the weblets are based on a cost model that accounts for various parameters, such as energy consumption, application performance and data privacy. Considering the pitfalls, the proposed prototype uses a simple weblet launch scheduling that does not truly reflect the effectiveness of the proposed cost model. The sharing of data and states between the weblets that execute on distributed locations are prone to security issues. Therefore, Zhang *et al.* [98] critically analyzed elastic applications for various security threats, such as authentication, trustworthiness (of the weblet containers), authorization, communication, and auditing. Nevertheless, the proposed model [97] is also affected by data sharing delays (smartphone-weblet, weblet-weblet) for which data replication solutions may be required. However, the data replication may give rise to data synchronization and integrity issues.

B. Energy Based Application Models

Energy based application models are designed to reduce energy consumption of mobile device applications by utilizing cloud resources. This is achieved by reducing the computational overhead of applications through computation offloading. Consequently, the resource intensive computational tasks are performed in the cloud and applications consume less energy on mobile devices.

1) μ Cloud: The μCloud [99] model focuses on the composition of applications from heterogeneous components to support flexibility, reusability, and configurability. Therefore, to achieve composition of applications from heterogeneous components, the application components are presented in the form of a graph, where each component may execute on a smartphone, cloud, or both (called hybrid components). The hybrid components may have multiple implementations and requires a middleware, such as WebOS [100], for execution. Moreover, the components are easily identifiable, and loosely bounded with input/output parameters, private memory and configuration information. In μCloud, the applications are presented as directed graphs, where the nodes represent components and the edges represent control flow between the components. Therefore, when an application graph executes, each component injects its output into the subsequent components. Figure 8 presents the application partitioning and execution model of the μCloud.

As shown in Figure 8, first an application is partitioned into small sub partitions (PA and PB), where each partition has components with homogeneous resource requirements. Further, the partitions are divided into fragments (A1, B1, B2) that are executed by the *orchestrator* in a many-to-many relationship. The architecture of an orchestrator is shown in Figure 9.

The orchestrator consists of three main elements, (1) a *Conductor* that executes components according to the application

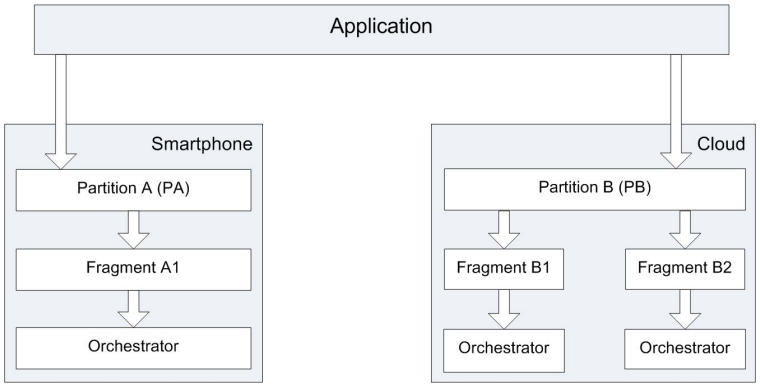

Fig. 8. Execution Model of μCloud [99]

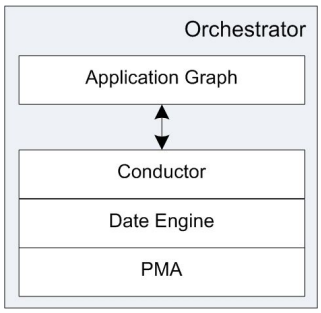

Fig. 9. Orchestrator architecture [99]

graph and passes the execution flow to other components, (2) a *data engine* that assists in optimizing the dataflow between the components, and (3) a *Performance Monitoring Agent (PMA)* that is responsible for monitoring the performance of an application.

The positive point of μCloud is that it supports self-contained application components that are decoupled from each other. μCloud requires skilled programmers for the development of the application components that are later used by the layman users for the development of applications. Consequently, the decoupled application components are reusable and provide flexibility in terms of application modeling. The negative point of the model is that it requires high programming efforts for the development of components. Moreover, in μCloud a single application partition can only execute on one orchestrator at a time. Nevertheless, no mechanism is proposed for securing the data that is exchanged between the components, making the model vulnerable to security and privacy threats.

C. Constraint Based Application Models

Constraint based application models are designed to execute applications in resource constrained environment (such as smartphones) by using cloud resources. For instance, consider a mobile device that has insufficient local resources (unavailable or overloaded) for execution of an application. In these models, the light weight application components execute on mobile device while the resource intensive components execute in the cloud. Consequently, these models enable high resource-demanding applications to execute on resource constrained devices.

1) Satyanarayanan et al. Model: In [2], Satyanarayanan *et al.* propose a model that is based on augmented execution technique. The model uses a concept of virtual machine that runs on trusted and resource-rich computer, or a cluster of computers named *cloudlet*. Moreover, the mobile devices act like a thin client, and offload resource-intensive tasks to the cloudlet. The article presents two approaches for the computation offloading, i.e., VM migration and VM synthesis [101]. The *VM migration* approach suspends the VM execution and saves the processor, disk and memory states. Next, the VM migrates to the cloudlet and resumes execution from the saved point. The feasibility of VM migration is supported by SoulPad [102], Collective [103], Internet Suspend/Resume (ISR) system [104], [105], and Xen live migration [106]. Alternatively, *VM synthesis* derives a small VM overlay from the mobile device and moves the overlay to the cloudlet. The VM overlay applies on the base VM and the execution resumes from the saved point, as shown in Figure 10.

The latency issue is very crucial in mobile cloud application models [107]. Therefore, the proposed model offloads the computation to the nearby infrastructure instead of distant clouds to avoid delays incurred by wide-area networks. Moreover, the mobile devices rely on low-latency, one-hop cloudlet that is accessible via a Wi-Fi connection. Also, the model supports parallelism that can be achieved by using the technique mentioned in [108].

The main advantage of Satyanarayanan *et al.* model is that the VM based approach is less fragile compared to the process migration and software virtualization [109]. This approach is also less restrictive in terms of language-based virtualization, where systems are bound to support specific programming languages. Consequently, if the cloudlet is a cluster, then VM parallelism can be achieved by using multiple cores [108]. Among the pitfalls of the model is that the VM synthesis process requires sixty to ninety seconds that makes the technique unsuitable for real-time tasks [2]. Moreover, the overlay extraction and compression that are performed on the smartphone requires computation and consumes battery power. Furthermore, if the smartphone VM overlay is from an old version base VM, then the overlay may not find a compatible cloudlet. Therefore, update patches are required for the old operating systems to make the overlays compatible with all cloudlets. Although the patches may resolve the

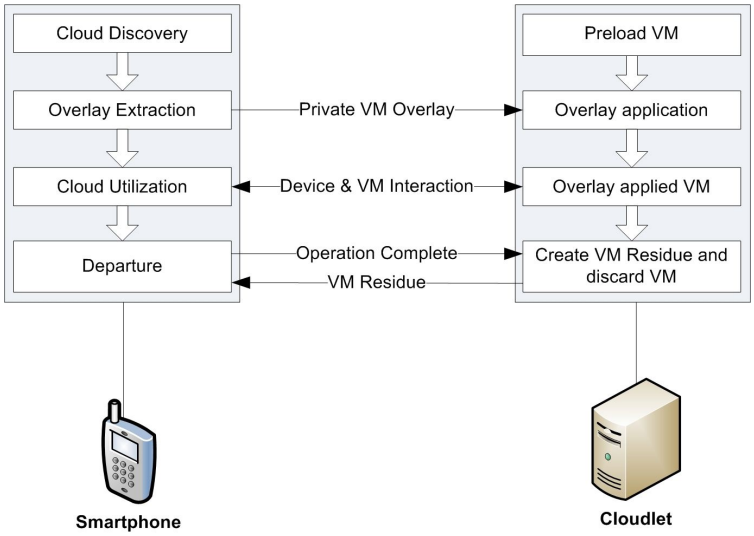

Fig. 10. VM Synthesis [2]

compatibility issue, the patches increase the size of the overlay. Alternatively, new replacement overlays may be required using update patches. The model also requires trust establishment schemes such as [110] and [111] to keep the users secure from malicious VMs. Lastly, the cloudlets are not available everywhere that makes the proposed model less scalable.

2) Giurgiu et al. Model: Giurgiu *et al.* [43] propose a model that focuses on partial offloading of the applications to the cloud/server. The proposed model is based on distributed layers technique in which functional layers are distributed between the smartphone and the server to optimize latency, data transfer delay, and cost. The model uses R-OSGi [112] and AlfredO [113] frameworks for the management and deployment of applications. R-OSGi is an enhanced version of OSGi [114] that supports multiple VMs residing on distributed servers, whereas the primary objective of OSGi is to assist with the decomposition and coupling of applications into modules, called *bundles*. Alternatively, AlfredO facilitates the distribution of bundles of multiple layers (presentation, logic and data) between the smartphone and the server. The presentation layer resides on the smartphone while the logic layer is distributed between the server and the smartphone. Moreover, the data layer is fully deployed on the server to minimize the data access delay. The architecture of AlfredO is represented in Figure 11.

The AlfredO system consists of AlfredOClient, AlfredOCore, and a renderer. The *AlfredOClient* fetches application bundles and services while the *AlfredOCore* is responsible for optimal deployment of bundles by using application partitioning algorithms such as *All-step* and *K-step*. In all-step algorithm, partitions are computed offline on the basis of phone hardware resources and network conditions. Conversely, in K-step algorithm the partitions are computed on run-time when a phone connects the server and specifies the hardware resources. Lastly, the *renderer* generates UI according to the application description that is received from the AlfredOCore.

In Giurgiu *et al.*'s model, the usage of applications is very unique. First, a connection is established to the server by R-OSGi with or without the involvement of SLP [115],

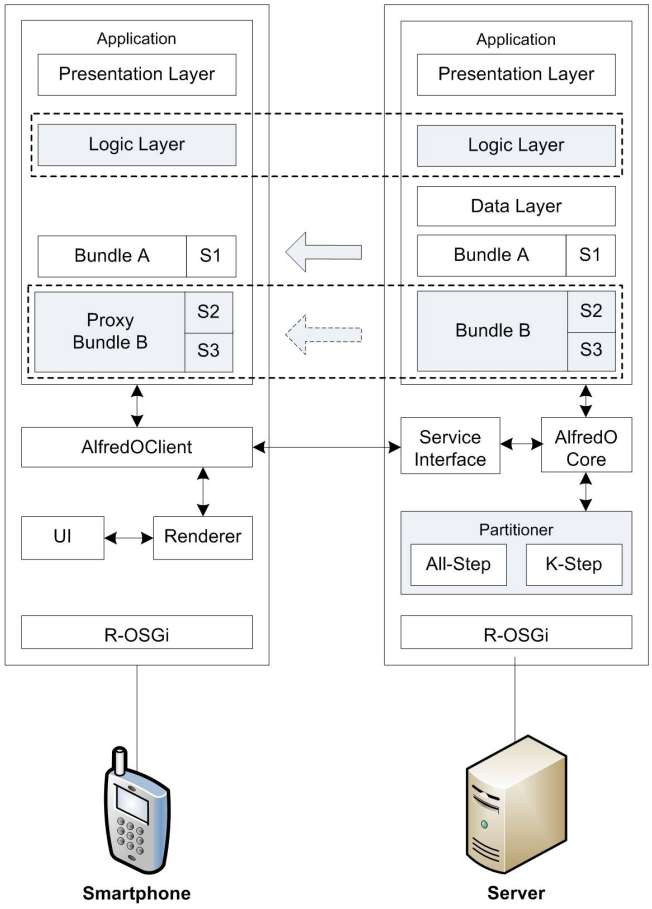

Fig. 11. AlfredO architecture

depending on whether the address is known or unknown. Then, AlfredOCore computes the optimal deployment of the bundles, and returns the application description and list of services to the AlfredOClient. Finally, the renderer generates a user interface according to the received application descriptor, and AlfredOClient fetches the required services. In this model, if a client wants to use a service that can run on the client,

then the server transfers the application bundle to the client as illustrated in Figure 11 (Bundle A - S1). However, if the service is bound to run on the server, then a client creates a local proxy to the server to use the service, as shown in Figure 11 (Bundle B proxy - S2 S3).

The main advantage of Giurgiu *et al.* model is that it supports heterogeneous client side environments and distributes functional layers between the smartphone and the server to optimize latency. However, as the user interface and service logic are tightly coupled, the modularization at the service logic level may involve changes in the user interface. Another shortcoming of Giurgiu *et al.* model is that the decision of the application component (bundle) distribution is server dependent that is not favorable in terms of server scalability and smartphone dynamic resource requirements. Consequently, a decision about component distribution that seems favorable at first, may turn unfavorable due to network conditions or smartphone resource availability. Therefore, timely reassessment of the (distributed) components is required, which may add to the overhead of the server and the smartphone. Moreover, due to the dependence on R-OSGI, higher delays are expected in proxy settings and component installation. Furthermore, the proposed model requires modifications in the application source code and increases burden on programmers.

3) eXCloud: eXCloud (Extensible Cloud) [116] supports VM instance level computation offloading to the cloud. eXCloud uses Stack-On-Demand (SOD) on top of VM systems to migrate the top stack frames or segments of the frames to the cloud. The code and heap data is left on the smartphone that is transferred later on demand. Moreover, eXCloud uses SOD Execution Engine (SODEE) layer between the applications and underlying components. SODEE layer is transparent to the applications, and no modifications are required in the application executables or JVM. Therefore, eXCloud migrate the tasks to the cloud when smartphone load exceeds certain level or the tasks are unable to execute on the smartphone due to insufficient resources. eXCloud also supports locality driven migration, where the computation is moved near to the data source to minimize data access delay. Figure 12 illustrates the main architecture of eXCloud. The main components of eXCloud are class pre-processor, migration manager, object preprocessor, communication manager and resource manager. The class preprocessor is responsible for adding *state capturing* and *restoring code* to the Java applications' byte code before it is loaded to the JVM. The *migration manager* serves the migration requests, while the *object pre-processor* handles the synchronization of objects among execution sites. The *worker manager* is responsible for the creation and management of worker processes. To offload a task to the cloud, an instance of the worker process is created in the cloud to receive and execute the offloaded task. Worker processes can be created in advance and set to wait in a standby mode to reduce execution delay. Next, the *communication manager* manages the communication between the cloud node and the smartphone. On the smartphone side, the *resource manager* is responsible for the provisioning of resources to the applications. Therefore, when resources are insufficient or unavailable, the resource manager requests the *migration manager* to perform migration of the task to the cloud where required resources are available. Consequently, the migration manager performs the task migration and waits for the result. On completion of the task, the result is returned to the smartphone and the application continues its normal execution.

Considering the advantages of eXCloud, it transfers only the top stack frames, unlike the traditional process migration techniques in which full state migrations are performed. eXCloud does not have any specific runtime requirements and works with standard JVM and Java libraries. Moreover, eXCloud provides multi-level task mobility, and provides lightweight partial state migration among cloud nodes and smartphones. Among the shortcomings of eXCloud is that it is not context aware and computation offloading is done based on local resource availability. Therefore, eXCloud offloads computation to the cloud whenever smartphone resources are insufficient or overloaded without considering energy and performance gain that can be achieved by offloading. Moreover, eXCloud does not take into account the availability of required resources (in the cloud), such as processing power and memory. Furthermore, for large input size, the eXCloud based applications may incur high computational time in the cloud environment due to on-demand data sharing between the smartphone and the cloud.

D. Multi-objective Application Models

The purpose of these models is to achieve multiple objectives mainly performance and energy efficiency at the same time with a fair tradeoff between required objectives. These models are considered more affective as they support multiple objectives unlike the performance and energy based models (discussed in A and B) that are designed to achieve a singular objective and may sacrifice energy efficiency or performance, respectively.

1) MAUI: MAUI [14] provides fine-grained application code offloading with minimum programmer intervention. The main focus of this model is to minimize energy consumption of mobile devices, which is the foremost challenge of the mobile industry. Therefore, MAUI offloads all the resource-intensive methods to the nearby infrastructure or cloud, provided the offloading is beneficial in terms of energy.

MAUI uses a profiler (optimization engine) that analyzes energy consumption involved in the local and remote execution of the code. Moreover, MAUI profiles offload methods and use history-based approach to predict the execution time of a particular code. Therefore, if the remote execution is beneficial in terms of energy, then the code is offloaded to the nearby infrastructure. In MAUI, the application partitioning is dynamic and the offloading is done on the basis of methods instead of complete application modules to minimize the offloading delay. However, MAUI creates two versions of smartphone application, for local and remote execution using Microsoft .NET Common Language Runtime (CLR) [117]. The architecture of MAUI is shown in Figure 13.

In MAUI, the mobile device consists of three main components, i.e., solver interface, profiler and client proxy. The *solver interface* provides interaction with the **solver** (decision engine) and facilitates the offloading decision making. The *profiler* collects information regarding the application energy

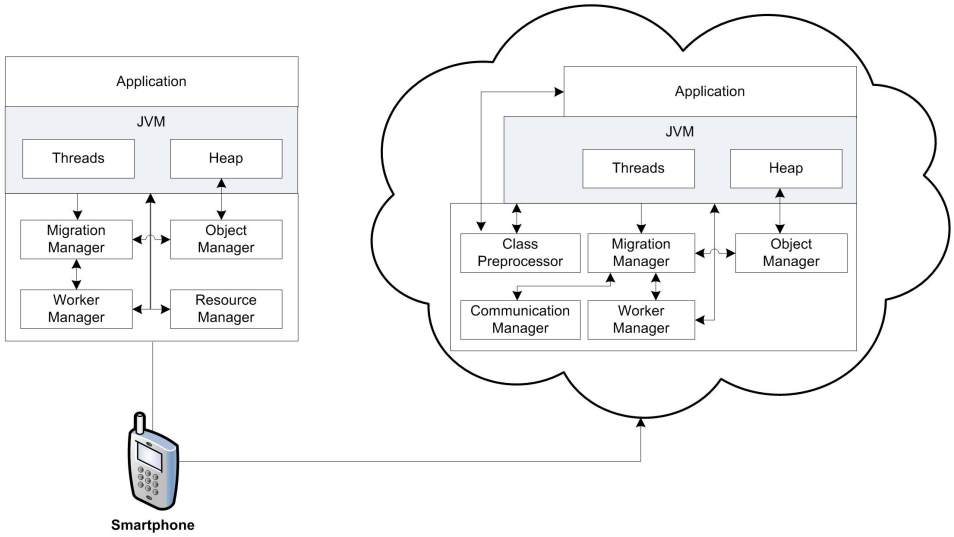

Fig. 12. eXCloud architecure

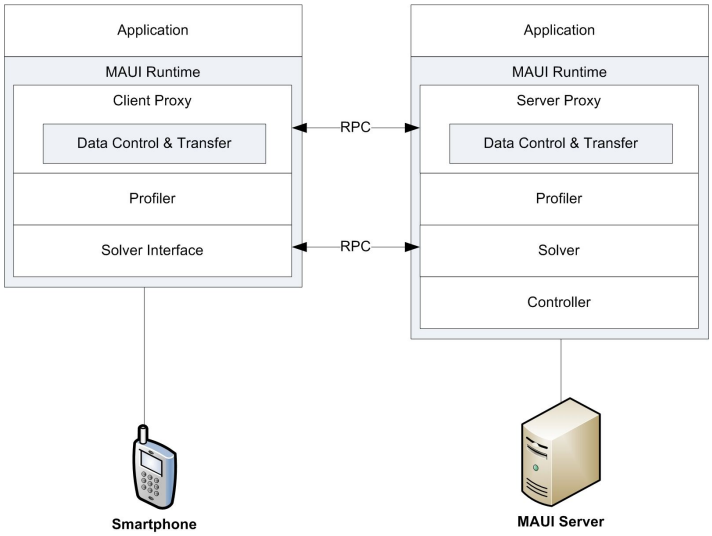

Fig. 13. MAUI architecture [14]

consumption and data transfer requirements. The *client proxy* deals with the method offloading and data transfer. Similarly, the server side consists of profiler, server proxy, solver and controller. However, the working of a profiler and server proxy is similar to the smartphone. The solver is the main decision engine of the MAUI that holds the call graph of the applications and the scheduled methods. Lastly, the *controller* is responsible for the authentication and resource allocation for incoming requests.

Considering the advantages, MAUI provides a programming environment where independent methods can be marked for remote execution. It uses dynamic partitioning of the applications to reduce burden on the programmers. Moreover, MAUI does not only focus on memory constrains of the smartphone but also considers the energy consumption involved in the offloading procedure. Furthermore, MAUI supports fine-grained method level offloading that can offload even single methods instead of offloading the whole software blocks. However, single method offloading is less beneficial compared to combined methods (multiple methods) offloading. Another weakness of MAUI is that if the programmer forgets to mark methods (for remote execution), MAUI will not be able to offload those methods. Also, MAUI saves information about the offloaded methods (for future decisions) and uses online profiling to create an energy consumption model. When new offloading requests are received, MAUI uses history data to predict the execution time of the task. However, the execution time of the task is input size dependant that is not considered by the MAUI. Therefore, the predictions of MAUI might be wrong, resulting in wrong offloading decisions. Nevertheless, the MAUI profilers consume processing power, memory and energy, which is an overhead on the smartphones.

2) ThinkAir: ThinkAir [44] supports method-level offloading to a smartphone clone executing in the cloud. It is designed to achieve the desired QoS by executing multiple clones of the smartphone in parallel [60]. ThinkAir requires minor

modifications in the source code of the applications. Therefore, it is the duty of the programmers to identify all resource-intensive methods that can be offloaded to the cloud for remote execution. When a remoteable (offloadable) method is called, ThinkAir starts the *profilers* to monitor the remoteable method and store the information for future offloading decisions. Moreover, the *execution controller* makes a decision about the execution location of the method that is based on the execution time, energy consumption, and previous execution history kept by the profilers.

For remote execution, ThinkAir sends the smartphone side calling object to the application server in the cloud, and waits for the result. If the connection to the cloud is disrupted, then ThinkAir falls back to local execution while attempting to reconnect asynchronously. Alternatively, if the method fails to execute in the cloud, then the exception is propagated back to the smartphone so that the local control flow of an application is not disturbed. Figure 14 illustrates the main architecture of the ThinkAir framework.

On the smartphone side, ThinkAir consist of an execution controller, a client handler, and profilers. The execution controller is responsible for the identification of remotable methods that are marked by the programmer. Moreover, the execution controller makes the offloading decisions and communicates with the server. The *client handler* is responsible for the execution of the communication protocols and management of the connection between the client and the cloud. The profilers are the most important part of the framework because the offloading decision is based on the accuracy of these profilers. Currently, ThinkAir supports three profilers that coordinate with the energy model. The *device profiler* monitors the energy consumption of the device hardware resources, such as processor, antennas, display screen etc. The *program profiler* monitors the program parameters, for instance, execution time, acquired memory, thread CPU time, number of instructions and method calls. Lastly, the *network profiler* monitors network related parameters, for instance, bandwidth, connectivity and delay.

On the server side, ThinkAir consist of a server handler, application server, and dynamic object input stream. The *server handler* is responsible for receiving computational requests and reporting the results back to the clients. It is also responsible for providing additional resources to support parallelism. The *application server* is responsible for the management of offloaded code. Besides, it is light-weight and facilities the process of replication. Lastly, the *dynamic object input stream* handles the exceptions generated during the execution of the offloaded code.

The main advantage of ThinkAir is that it takes into account the energy consumption when making the offloading decisions, and supports on-demand resource allocation and parallelism to reduce execution delays. The model offloading decisions are based on the profilers, and uses energy model to estimate energy consumption. ThinkAir's energy model is inspired by PowerTutor [118] that accounts all parameters of the supported profilers. Nevertheless, it does not require separate application servers for the distribution of the applications. Considering the shortcomings, ThinkAir does not support unmodified applications and requires programmers support for the demarcation of offloadable methods. Therefore, if any offloadable methods are left unmarked, then ThinkAir will not be able to offload those methods, which may affect the performance of the applications. Nevertheless, the profiling process of the model incurs an overhead on the smartphone because it consumes computation power, memory and energy.

3) Cuckoo: Cuckoo [119] is based on partial offloading of the applications to the cloud/nearby infrastructure, and is designed with the objective to make the programming easy for the developers by integrating the existing development tools that are familiar to the developers. Moreover, Cuckoo is designed for Android platform [89] and supports both local and remote method implementations. Figure 15 illustrates the Cuckoo application development process and model architecture.

For Cuckoo application development, a developer creates a project and writes the source code. Next, by using the existing activity/service model of Android [120], computation intensive (services) and interactive parts (activities) of the application are separated. The separation is done with the help of an interface definition language called AIDL [121]. Further, the build system creates an interface and remote service that contains a dummy implementation done by the Cuckoo Remote Service Deriver (CRSD). Moreover, a stub/proxy is generated by the Cuckoo Service Rewriter (CSR) for each AIDL interface so that a method can be invoked locally or remotely on the basis of information provided by the Cuckoo Resource Manager. Implementation is done as both local and remote service. The local and remote interface implementation may seem identical, but behaves differently, as the algorithms and libraries may vary according to the location of the service execution. Later, the developer codes the local service implementation and overwrites the remote service implementation by using CSR. Finally, the build system compiles the code and provides an installable .apk file to the users.

The Cuckoo based applications can offload their computation to any Java Virtual Machine (JVM) residing on the nearby infrastructure or cloud. Therefore, it is the smartphones' application responsibility to install the service(s) on the server. Once the service is installed, the address of the server is passed to the resource manager running on the smartphone in the form of two dimension barcode [122] or resource description file. Finally, the address registrar registers the address and the remote resource becomes usable for the smartphone applications.

The main advantage of Cuckoo is that it supports partial offloading of the applications to the cloud and uses well known tools for application development. Considering the shortcomings, Cuckoo does not support asynchronous callbacks and state transferring from remote resources. Moreover, no states are saved while transferring from local to remote execution or vice versa for which Representational State-Transfer (REST) [123] may be required. Another shortcoming of Cuckoo is that it requires programmers support for the modification of applications. Furthermore, it lacks security features to restrict users from installing malfunctioned codes on the server and control illegal access to the resources. Nevertheless, the offloading decisions of Cuckoo are static and context unaware.

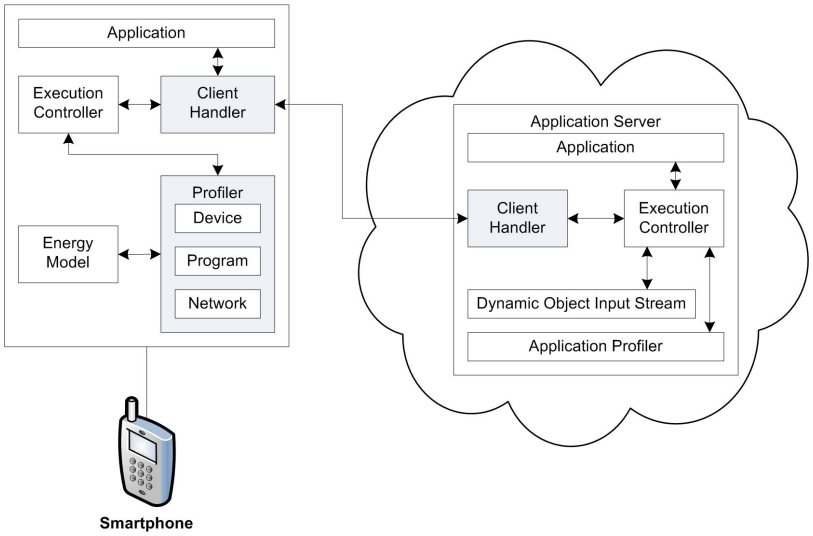

Fig. 14. ThinkAir framework architecture [44]

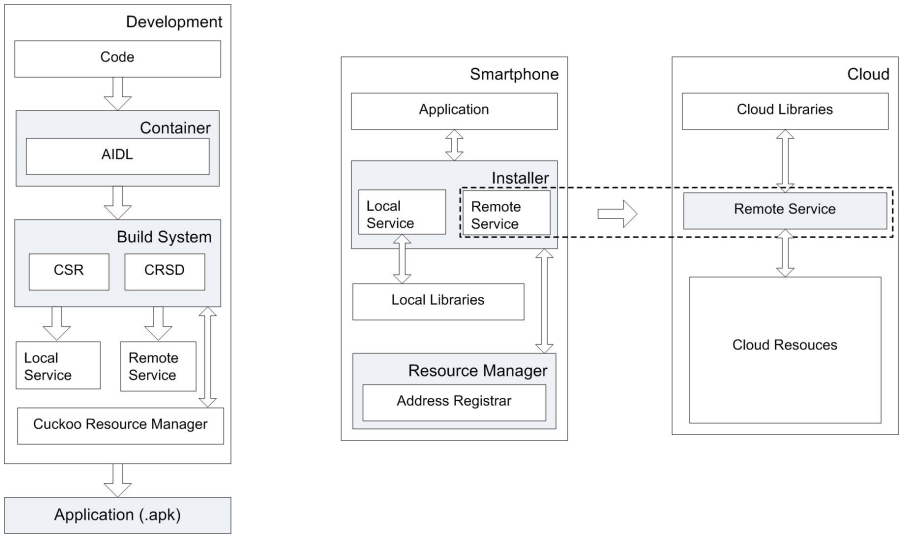

Fig. 15. Cuckoo architecture and application development process

V. Models Comparisson and Application Examples

From the analysis presented in Section IV, it is evident that none of the application models have considered all parameters highlighted in Section III. All models except [97] have ignored the security and privacy issues of the applications' data, and smartphones' clone that resides in the cloud for augmented execution. Moreover, none of the application models have focused the application piracy control in the cloud environment.

A. Application Models Comparison

Table II presents the comparison of application models (discussed in Section IV) according to criteria mentioned in Section III.

Table III presents a comparison of nearby infrastructure and public cloud platform according to criteria mentioned in Section III.

B. Mobile Cloud Computing Application Examples

Mobile devices can benefit from mobile cloud computing in multiple perspectives as discussed in section IV. Therefore, mobile cloud computing can facilitate multiple domains. For the proof of concept, researchers have implemented many applications using different application models (listed in Table II). Here, we present few more mobile cloud computing application examples to show how this technology can prove beneficial for various domains.

- *Mathematical Tools:* Complex mathematical calculations [132], such as multiplication of a very large matrix require large computations and consume considerable amount of energy. Therefore, mobile devices can offload such computations to cloud that may help to gain energy efficiency and enhanced performance due to high computational power of the cloud. This type of applications can be developed using MAUI [14], ThinkAir [44], μCloud [99], eXCloud [116], and Cuckoo [119] model.

TABLE II
COMPARISON OF THE APPLICATION MODELS

Model	Ca	La	Bu	Ge	Pr	Co	Se	Pa	Sc	Er	Pt	Ta	Mc
CloneCloud [95]	M	L/ H/ M	H	M	L	H	L	H	L	NI/ CL	DalvikVM (Android)	Virus scan/ image search	P
Zhang *et al.* Model [97]	M	M	L	H	M	L	H	H	H	CL	.NET (C ♯)	Image processing	P
μCloud [99]	L	L	L	L	L	H	L	H	L	CL	Android	Face recognition	E
Satyanarayanan *et al.* Model [2]	L	H	M/ H	L	L	M	L	M	L	NI	VirtualBox (VMM for Linux)	Applications [124], [125], [126], [127], [128]	C
Giurgiu *et al.* Model [43]	L	L	L	M	M	M	L	H	M	NI	OSGI (Java)	Sweet Home 3D [129]	C
eXCloud [116]	L	M/ H	M	M	M	H	L	H	H	NI/ CL	iOS (JamVM)	Mathematical Calculations [130], [131]	C
MAUI [14]	H	L	L	H	L	L	L	H	L	NI/ CL	Microsoft .Net	Face recognition, games	MO
ThinkAir [44]	H	L	L	H	L	H	L	L	M	CL	NDK (Java)	N-queens problem [131]	MO
Cuckoo [119]	L	M	L	H	L	M	L	L	H	NI/ CL	Android	Object/face recognition	MO

Ca: Context Awareness
La: Latency
Bu: Bandwidth Utilization
Ge: Generality
Pr: Privacy
Co: Complexity
Se: Security

Pa: Programming Abstraction
Sc: Scalability
Er: Execution Resource
Pt: Platform
Ta: Tested/ Proposed Application
Mc: Model Category

CL: Cloud
NI: Nearby Infrastructure
H: High, M: Medium, L: Low
P: Performance Based
E: Energy Based
C: Constraint Based
MO: Multi-Objective Based

TABLE III
COMPARISON OF NEARBY INFRASTRUCTURE AND CLOUD PLATFORM

	Nearby Infrastructure (Virtual Cloud)	Real Cloud
Programming Abstraction	Neutral	Neutral
Scalability	Low	High
Security	Favorable	Favorable/ Unfavorable
Bandwidth Utilization	Neutral	Neutral
Latency	Favorable	Unfavorable
Complexity	Neutral	Neutral
Generality	Neutral	Neutral
Privacy	Favorable	Unfavorable
Context awareness	Neutral	Neutral

- *File Search:* The latest smartphones have storage capacity of up to 80 gigabytes. Therefore, smartphones can store large amount of files due to which search functions may take up to few minutes. However, by using mobile cloud computing the search functions can execute on the smartphone clone (in the cloud) that may result as enhanced performance and energy efficiency (assuming that the smartphone and smartphone clone in the cloud are pre-synchronized) [83]. This type of applications can be developed using Satyanarayanan *et al.* [2], and CloneCloud [95] Model.
- *Imaging Tools:* Image processing tasks demand large computations and the operations may take up to a few minutes for completion, for instance, when rendering a 3D image from a source file [133]. Therefore, the imaging tools can offload heavy computational operation to the cloud and benefit from enhanced performance and energy efficiency depending on the available resources (network, cloud) and runtime conditions. Such applications can be developed using MAUI [14], ThinkAir [44], μCloud [99], eXCloud [116], and Cuckoo [119] model.
- *Games:* Gaming applications usually require heavy computation (on large datasets) and quick response time for user interaction. Therefore, computation offloading is not recommended as it may reduce the game performance. For instance, First Person Shooting (FPS) games [134] are not suitable for computation offloading. However, some games require large computation using small datasets that enable quick computation offloading and lead to energy efficiency, for instance, chess game [14]. This type of games can be developed using MAUI [14], ThinkAir [44], μCloud [99], eXCloud [116], and Cuckoo [119] model.
- *Download Applications:* Downloading files at low data rate consumes high energy compared to high data rate. Therefore, it is beneficial in terms of energy to download files in the cloud and then transfer files to the mobile device with high speed. For instance, mobile cloud BitTorrent application [135] can download file parts from multiple peers in the cloud, and then transfer the download file to mobile device at high speed to reduce energy consumption. Such applications can be developed using Zhang et al's model [97].
- *Antivirus Applications:* Considering the increasing threat from viruses and malwares, antivirus applications are becoming vital part of smartphones. However, scanning a smartphone for viruses requires computation that consume high amount of energy. Using mobile cloud

computing, the smartphone clone can be scanned in the cloud to save energy (assuming that the smartphone and smartphone clone in the cloud are pre-synchronized) [44]. This type of applications can be developed using Satyanarayanan *et al.* [2], and CloneCloud [95] Model.

- *Security:* The software and hardware level enhancements of smartphones enable them to execute wide range of applications. However, installing large number of applications may increase the threat from malwares that can jeopardize users' personal information stored on the smartphone. Using mobile cloud computing, the mobile device applications can execute their services in the cloud that may reduce the threat from attackers. Consequently, the mobile devices execute more trusted and less complex applications with enhanced security and energy efficiency. The aforementioned security aspects is the focus of the MobiCloud framework [136] along many other important issues, such as trust management and secure routing.

There can be hundreds of similar applications that can take advantage of mobile cloud computing. However, the objective of using mobile cloud computing (such as energy efficiency, performance enhancement, and application execution) must be known before adopting any application model. For example, the chess game can offload computation to the cloud at runtime to save energy. Alternatively, for file search, energy efficiency is only achieved when the smartphone clone is already available in the cloud and synchronized with the smartphone. Therefore, application models must be chosen wisely to achieve the desired objective(s).

Nevertheless, mobile cloud applications require variable amount of communication for computation offloading that depends on the nature of application and models' working (as discussed in Section IV). It is a fact that mobile cloud applications will increase the overall Internet traffic and threaten the revenue of mobile network operators in the upcoming years [137]. However, mobile cloud applications open new ways of effective communication that can help to reduce data communication between the mobile devices and network operators. Consider the aforementioned torrent example in which required file is downloaded in the cloud and then transferred to mobile device with high speed. Consequently, the mobile network communication channel is utilized affectively and for limited time. Similarly, efficient communications of cellular radios are great contributors of energy efficiency on mobile devices. Therefore, mobile cloud applications can also help to reduce energy consumption by using aggregation and compression techniques in the cloud to achieve energy efficiency on mobile devices [138]. Nevertheless, mobile cloud applications may help to secure communication [136] and facilitate routing [139] by using cloud technology.

VI. Conclusion

A number of the application models discussed in Section IV impose intensive coding on the programmers. In order to ease the burden on the programmers, new programming tools are required that provide programming abstraction and hide the underlying complexities of the cloud and mobile devices. The developed applications usually support one execution platform, thus, limiting the offloading of the elements (applications, components, clones) to other platforms. The mobile cloud execution platforms need to be standardized to ease computation offloading to the mobile cloud platforms. Also, new energy consumption models are required to facilitate accurate decision making by considering the main entities involved in the offloading process.

The mobile cloud application models that are based on augmented execution of the smartphone clone in the cloud require synchronization of the smartphone and the clone. Therefore, new synchronization policies are required that can perform timely synchronization, taking into account accuracy, execution delay, and bandwidth utilization. Moreover, a smartphone clone contains its user's data and licensed applications that are vulnerable to security attacks and piracy issues. A security mechanism is required to secure the clones from illegal access and protect the smartphone users from the malicious VMs executing in the cloud. Nevertheless, if a smartphone clone falls into the wrong hands, then the adversary may install the clone on a smartphone of the same model and access the licensed applications illegally. To handle this issue, a new mobile cloud application piracy control framework is required.

Some European Union data management laws and cloud computing principals are contrary to each other. Moreover, the provision of virtually unlimited resources to untrustworthy users may cause problems for the victims (enterprises, users) and the service providers. Therefore, new policies are required that can confine mobile user access to optimum resources, or timely identify and revoke access of the untrustworthy users. Consequently, there is a need to standardize the mobile cloud computing platforms and refine the data management laws accordingly, so that the mobile cloud computing can flourish and mobile users can truly benefit from the cloud computing technology.

References

[1] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets," *IEEE Commun. Surveys & Tutorials*, vol. 99, pp. 1–20, 2012.

[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[3] C. Mascolo, "The power of mobile computing in a social era," *IEEE Internet Computing*, vol. 14, no. 6, pp. 76–79, 2010.

[4] E. Barba, B. MacIntyre, and E. D. Mynatt, "Here we are! where are we? locating mixed reality in the age of the smartphone," *Proc. IEEE*, vol. 100, no. 4, pp. 929–936, 2012.

[5] A. Wright, "Get smart," *Commun. ACM*, vol. 52, no. 1, pp. 15–16, 2009.

[6] R. Kemp, N. Palmer, T. Kielmann, F. Seinstra, N. Drost, J. Maassen, and H. Bal, "eyedentify: Multimedia cyber foraging from a smartphone," in *Multimedia, 2009. ISM'09. 11th IEEE International Symposium on*. IEEE, 2009, pp. 392–399.

[7] Amazon simple storage service. Accessed December 8th, 2011. [Online]. Available: http://aws.amazon.com/s3/

[8] X. Yang, T. Pan, and J. Shen, "On 3g mobile e-commerce platform based on cloud computing," in *Ubi-media Computing (U-Media), 2010 3rd IEEE International Conference on*. IEEE, 2010, pp. 198–201.

[9] C. Doukas, T. Pliakas, and I. Maglogiannis, "Mobile healthcare information management utilizing cloud computing and android os," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. IEEE, 2010, pp. 1037–1040.

[10] W.-T. Tang, C.-M. Hu, and C.-Y. Hsu, "A mobile phone based home-care management system on the cloud," in *Biomedical Engineering and Informatics (BMEI), 2010 3rd International Conference on*, vol. 6. IEEE, 2010, pp. 2442–2445.

[11] R. Ferzli and I. Khalife, "Mobile cloud computing educational tool for image/video processing algorithms," in *Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), 2011 IEEE*. IEEE, 2011, pp. 529–533.

[12] W. Zhao, Y. Sun, and L. Dai, "Improving computer basis teaching through mobile communication and cloud computing technology," in *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, vol. 1. IEEE, 2010, pp. V1–452.

[13] Facebook. Accessed November 26th, 2011. [Online]. Available: http://facebook.com

[14] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proc. 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 49–62.

[15] Flickr. Accessed December 26th, 2011. [Online]. Available: http://flickr.com

[16] V. S. Pendyala and J. Holliday, "Performing intelligent mobile searches in the cloud using semantic technologies," in *Granular Computing (GrC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 381–386.

[17] D. Huang, "Mobile cloud computing," *IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter*, vol. 6, no. 10, pp. 27–31, 2011.

[18] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in *Proc. 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*. ACM, 2010, p. 6.

[19] T. Xing, D. Huang, S. Ata, and D. Medhi, "Mobicloud: A geo-distributed mobile cloud computing platform," in *Network and Service Management (CNSM), 2012 8th International Conference on*. IEEE, 2012, pp. 164–168.

[20] T. Xing, H. Liang, D. Huang, and L. X. Cai, "Geographic-based service request scheduling model for mobile cloud computing," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 1446–1453.

[21] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," *IEEE Network*, vol. 24, no. 4, pp. 19–24, 2010.

[22] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Trans. Computer Systems (TOCS)*, vol. 26, no. 2, p. 4, 2008.

[23] S. Sakr, A. Liu, D. M. Batista, and M. Alomari, "A survey of large scale data management approaches in cloud environments," *IEEE Commun. Surveys & Tutorials*, vol. 13, no. 3, pp. 311–336, 2011.

[24] X. Fan, J. Cao, and H. Mao, "A survey of mobile cloud computing," *ZTE Communications*, vol. 9, no. 1, pp. 4–8, 2011.

[25] E. E. Marinelli, "Hyrax: cloud computing on mobile devices using mapreduce," DTIC Document, Tech. Rep., 2009.

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

[27] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, 2011.

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

[29] D. Kovachev, Y. Cao, and R. Klamma, "Mobile cloud computing: a comparison of application models," *arXiv preprint arXiv:1107.4940*, 2011.

[30] Rackspace cloud. Accessed April 10th, 2012. [Online]. Available: http://www.rackspace.com/

[31] Google app engine. Accessed November 15th, 2011. [Online]. Available: http://appengine.google.com

[32] Google apps for business. Accessed April 10th, 2012. [Online]. Available: http://www.google.com/enterprise/apps/business/

[33] Salesforce. Accessed April 10th, 2012. [Online]. Available: http://www.salesforce.com/cloudcomputing/

[34] Amazon elastic compute cloud (ec2),. Accessed December 10th, 2011. [Online]. Available: http://www.amazon.com/ec2/

[35] Microsoft azure. Accessed April 10th, 2012. [Online]. Available: http://www.windowsazure.com

[36] Integrated cellular satellite solution. Accessed November 26th, 2012. [Online]. Available: https://www.wireless.att.com/businesscenter/business-programs/government/solutions/integrated-cellular-satellite-solution.jsp

[37] Spot connect. Accessed November 27th, 2012. [Online]. Available: http://www.findmespot.com/en/index.php?cid=116

[38] Sony xperia s. Accessed May 15th, 2012. [Online]. Available: http://www.gsmarena.com/sony_xperia_s-4369.php

[39] Htc one x. Accessed May 15th, 2012. [Online]. Available: http://www.gsmarena.com/htc_one_x-4320.php

[40] Vmware. Accessed April 10th, 2012. [Online]. Available: http://www.vmware.com/

[41] Virtualbox. Accessed April 10th, 2012. [Online]. Available: http://www.virtualbox.org/

[42] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[43] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: enabling mobile phones as interfaces to cloud applications," in *Middleware 2009*. Springer, 2009, pp. 83–102.

[44] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Unleashing the power of mobile cloud computing using thinkair," *arXiv preprint arXiv:1105.3232*, 2011.

[45] A. Ghosh, R. Ratasuk, B. Mondal, N. Mangalvedhe, and T. Thomas, "Lte-advanced: next-generation wireless broadband technology [invited paper]," *Wireless Communications, IEEE*, vol. 17, no. 3, pp. 10–22, 2010.

[46] Glympse. Accessed November 26th, 2011. [Online]. Available: https://market.android.com/details?id=com.glympse.android.glympse

[47] Locale. Accessed November 26th, 2011. [Online]. Available: https://market.android.com/details?id=com.twofortyfouram.locale

[48] Foursquare. Accessed November 26th, 2011. [Online]. Available: https://market.android.com/details?id=com.joelapenna.foursquared

[49] Google maps for android. Accessed November 26th, 2011. [Online]. Available: https://market.android.com/details?id=com.google.android.apps.maps

[50] Tape-a-talk voice recorder. Accessed November 26th, 2011. [Online]. Available: https://market.android.com/details?id=name.markus.droesser.tapeatalk

[51] Sqllite. Accessed November 26th, 2011. [Online]. Available: https://market.android.com/details?id=com.xuecs.sqlitemanager

[52] A. N. Khan, M. Mat Kiah, S. U. Khan, and S. A. Madani, "Towards secure mobile cloud computing: a survey," *Future Generation Computer Systems*, 2012.

[53] S. Wang and X. S. Wang, "In-device spatial cloaking for mobile user privacy assisted by the cloud," in *Mobile Data Management (MDM), 2010 Eleventh International Conference on*. IEEE, 2010, pp. 381–386.

[54] P. Murray, "Enterprise grade cloud computing," in *Proc. Third Workshop on Dependable Distributed Data Management*. ACM, 2009, pp. 1–1.

[55] Council of july 2002: Directive 2002/58/ec concerning the processing of personal data and the protection of privacy in the electronic communications sector. European Parliament. Accessed November 20th, 2011. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32002L0058:EN:NOT

[56] Council of oct. 1995: Directive 95/46/ec on the protection of individuals with regard to the processing of personal data and on the free movement of such data. European Parliament. Accessed November 20th, 2011. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:NOT

[57] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Grid Computing Environments Workshop, 2008. GCE'08*. IEEE, 2008, pp. 1–10.

[58] M. Jakobsson, E. Shi, P. Golle, and R. Chow, "Implicit authentication for mobile devices," in *Proc. 4th USENIX conference on Hot topics in security*. USENIX Association, 2009, pp. 9–9.

[59] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit authentication through learning user behavior," in *Information Security*. Springer, 2011, pp. 99–113.

[60] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, 2009.

[61] D. Vali, S. Paskalis, L. Merakos, and A. Kaloxylos, "A survey of internet QoS signaling," *IEEE Commun. Surveys & Tutorials*, vol. 6, no. 4, pp. 32–43, 2004.

[62] Y. Bai and M. R. Ito, "QoS control for video and audio communication in conventional and active networks: approaches and comparison," *IEEE Commun. Surveys & Tutorials*, vol. 6, no. 1, pp. 42–49, 2004.

[63] L. Hanzo and R. Tafazolli, "A survey of QoS routing solutions for mobile ad hoc networks," *IEEE Commun. Surveys & Tutorials*, vol. 9, no. 2 2nd Quarter, pp. 50–70, 2007.

[64] Z. Zhou and D. Huang, "Efficient and secure data storage operations for mobile cloud computing," in *Network and Service Management (CNSM), 2012 8th International Conference on*. IEEE, 2012, pp. 37–45.

[65] Y. Zhu, H. Hu, G.-J. Ahn, D. Huang, and S. Wang, "Towards temporal access control in cloud computing," in *INFOCOM, 2012 Proc. IEEE*. IEEE, 2012, pp. 2576–2580.

[66] M. Lima, A. Dos Santos, and G. Pujolle, "A survey of survivability in mobile ad hoc networks," *IEEE Commun. Surveys & Tutorials*, vol. 11, no. 1, pp. 66–77, 2009.

[67] D. Djenouri, L. Khelladi, and N. Badache, "A survey of security issues in mobile ad hoc networks," *IEEE Commun. Surveys & Tutorials*, vol. 7, no. 4, 2005.

[68] The future of cloud computing. opportunities for european cloud computing beyond 2010. Expert Group Report. Accessed November 10th, 2011. [Online]. Available: http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf

[69] Avg antivirus and internet security. Accessed November 25th, 2011. [Online]. Available: http://www.avg.com

[70] Kaspersky lab. Accessed November 25th, 2011. [Online]. Available: http://www.kaspersky.com

[71] Esat antivirus software. Accessed November 25th, 2011. [Online]. Available: http://www.eset.com

[72] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, "Virtualized in-cloud security services for mobile devices," in *Proc. First Workshop on Virtualization in Mobile Computing*. ACM, 2008, pp. 31–35.

[73] J. Ogness, "Dazuko: An open solution to facilitate on-access scanning," *Virus Bulletin*, 2003.

[74] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proc. 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 55–66.

[75] W. Itani, A. Kayssi, and A. Chehab, "Energy-efficient incremental integrity for securing storage in mobile cloud computing," in *Energy Aware Computing (ICEAC), 2010 International Conference on*. IEEE, 2010, pp. 1–2.

[76] A. S. Tanenbaum and M. Van Steen, *Distributed systems*. Prentice Hall, 2002, vol. 2.

[77] G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid Android: versatile protection for smartphones," in *Proc. 26th Annual Computer Security Applications Conference*. ACM, 2010, pp. 347–356.

[78] P. Makris, D. Skoutas, and C. Skianis, "A survey on context-aware mobile and wireless networking: On networking and computing environments' integration," *IEEE Commun. Surveys & Tutorials*, vol. 15, pp. 362–386, 2012.

[79] S. U. Khan, "A multi-objective programming approach for resource allocation in data centers," in *International conference on parallel and distributed processing techniques and applications (PDPTA)*, 2009, pp. 152–158.

[80] S. U. Khan, "A goal programming approach for the joint optimization of energy consumption and response time in computational grids," in *Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th International*. IEEE, 2009, pp. 410–417.

[81] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[82] A. Bialecki, M. Cafarella, D. Cutting, and O. OMALLEY, "Hadoop: a framework for running applications on large clusters built of commodity hardware," *Wiki at http://lucene. apache. org/hadoop*, vol. 11, 2005.

[83] B.-G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *Proc. 12th conference on Hot topics in operating systems*. USENIX Association, 2009, pp. 8–8.

[84] Microsoft live mesh. Accessed November 15th, 2011. [Online]. Available: http://www.mesh.com

[85] J. Kolodziej, S. U. Khan, and F. Xhafa, "Genetic algorithms for energy-aware scheduling in computational grids," in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on*. IEEE, 2011, pp. 17–24.

[86] S. U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 3, pp. 346–360, 2009.

[87] D. Kliazovich, P. Bouvry, and S. U. Khan, "Dens: data center energy-efficient network-aware scheduling," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*. IEEE, 2010, pp. 69–75.

[88] S. U. Khan and C. Ardil, "Energy efficient resource allocation in distributed computing systems," in *International conference on distributed, high-performance and grid computing*. Citeseer, 2009, pp. 667–673.

[89] Android. Accessed November 21st, 2011. [Online]. Available: http://www.android.com

[90] Apple ios5. Accessed November 21st, 2011. [Online]. Available: http://www.apple.com/ios

[91] Nokia symbian. Accessed November 21st, 2011. [Online]. Available: http://symbian.nokia.com

[92] Windows mobile. Accessed November 21st, 2011. [Online]. Available: http://www.microsoft.com/download/en/windowsMobile.aspx

[93] Blackberry 7 os. Accessed November 21st, 2011. [Online]. Available: http://www.rim.com/products/blackberry_os7.shtml

[94] Nokia strategy 2011. Accessed November 22nd, 2011. [Online]. Available: http://conversations.nokia.com/nokia-strategy-2011

[95] B.-G. Chun, S. Ihm, P. Maniatis, and M. Naik, "Clonecloud: boosting mobile device applications through cloud clone execution," *arXiv preprint arXiv:1009.3088*, 2010.

[96] M. Satyanarayanan, M. A. Kozuch, C. J. Helfrich, and D. R. OHallaron, "Towards seamless mobility on pervasive hardware," *Pervasive and Mobile Computing*, vol. 1, no. 2, pp. 157–189, 2005.

[97] X. Zhang, S. Jeong, A. Kunjithapatham, and S. Gibbs, "Towards an elastic application model for augmenting computing capabilities of mobile platforms," in *Mobile wireless middleware, operating systems, and applications*. Springer, 2010, pp. 161–174.

[98] X. Zhang, J. Schiffman, S. Gibbs, A. Kunjithapatham, and S. Jeong, "Securing elastic applications on mobile devices for cloud computing," in *Proc. 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 127–134.

[99] V. March, Y. Gu, E. Leonardi, G. Goh, M. Kirchberg, and B. S. Lee, "μcloud: towards a new paradigm of rich mobile applications," *Procedia Computer Science*, vol. 5, pp. 618–624, 2011.

[100] Hewlett packard, hp webos 2.0. Accessed December 2nd, 2011. [Online]. Available: http://www.palm.com/us/products/software/webos2

[101] A. Wolbach, J. Harkes, S. Chellappa, and M. Satyanarayanan, "Transient customization of mobile computing infrastructure," in *Proc. First Workshop on Virtualization in Mobile Computing*. ACM, 2008, pp. 37–41.

[102] R. Cáceres, C. Carter, C. Narayanaswami, and M. Raghunath, "Reincarnating pcs with portable soulpads," in *Proc. 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 65–78.

[103] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum, "Optimizing the migration of virtual computers," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 377–390, 2002.

[104] M. Kozuch and M. Satyanarayanan, "Internet suspend/resume," in *Mobile Computing Systems and Applications, 2002. Proc. Fourth IEEE Workshop on*. IEEE, 2002, pp. 40–46.

[105] M. Satyanarayanan, B. Gilbert, M. Toups, N. Tolia, D. R. O'Hallaron, A. Surie, A. Wolbach, J. Harkes, A. Perrig, D. J. Farber *et al.*, "Pervasive personal computing in an internet suspend/resume system," *IEEE Internet Computing*, vol. 11, no. 2, pp. 16–25, 2007.

[106] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

[107] H. A. Lagar-Cavilla, N. Tolia, E. De Lara, M. Satyanarayanan, and D. OHallaron, "Interactive resource-intensive applications made easy," in *Middleware 2007*. Springer, 2007, pp. 143–163.

[108] H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, E. De Lara, M. Brudno, and M. Satyanarayanan, "Snowflock: rapid virtual machine cloning for cloud computing," in *Proc. 4th ACM European conference on Computer systems*. ACM, 2009, pp. 1–12.

[109] S. Osman, D. Subhraveti, G. Su, and J. Nieh, "The design and implementation of zap: A system for migrating computing environments," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 361–376, 2002.

[110] S. Garriss, R. Cáceres, S. Berger, R. Sailer, L. van Doorn, and X. Zhang, "Trustworthy and personalized computing on public kiosks,"

in *Proc. 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 199–210.
[111] A. Surie, A. Perrig, M. Satyanarayanan, and D. J. Farber, "Rapid trust establishment for pervasive personal computing," *IEEE Pervasive Computing*, vol. 6, no. 4, pp. 24–30, 2007.
[112] J. S. Rellermeyer, G. Alonso, and T. Roscoe, "R-osgi: distributed applications through software modularization," in *Proc. ACM/IFIP/USENIX 2007 International Conference on Middleware*. Springer-Verlag New York, Inc., 2007, pp. 1–20.
[113] J. S. Rellermeyer, O. Riva, and G. Alonso, "Alfredo: an architecture for flexible interaction with electronic devices," in *Proc. 9th ACM/IFIP/USENIX International Conference on Middleware*. Springer-Verlag New York, Inc., 2008, pp. 22–41.
[114] O. Alliance, "Osgi service platform, core specification, release 4, version 4.1," *OSGi Specification*, 2007.
[115] J. Veizades and C. E. Perkins, "Service location protocol," 1997.
[116] R. K. Ma, K. T. Lam, and C.-L. Wang, "excloud: Transparent runtime support for scaling mobile applications in cloud," in *Cloud and Service Computing (CSC), 2011 International Conference on*. IEEE, 2011, pp. 103–110.
[117] J. Richter, *CLR via C#*. Microsoft Press, 2010.
[118] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. ACM, 2010, pp. 105–114.
[119] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," in *Mobile Computing, Applications, and Services*. Springer, 2012, pp. 59–79.
[120] Android application fundamentals. Accessed December 3rd, 2011. [Online]. Available: http://developer.android.com/guide/topics/fundamentals.html
[121] Android interface definition language (aidl). Accessed December 3rd, 2011. [Online]. Available: http://developer.android.com/guide/developing/tools/aidl.html
[122] Denso wave qr. Accessed December 3rd, 2011. [Online]. Available: http://www.denso-wave.com/qrcode/index-e.html
[123] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *ACM Transactions on Internet Technology (TOIT)*, vol. 2, no. 2, pp. 115–150, 2002.
[124] Abiword. Accessed December 1st, 2012. [Online]. Available: http://www.abisource.com
[125] Gimp. Accessed December 1st, 2012. [Online]. Available: http://www.gimp.org
[126] Gnumeric. Accessed December 1st, 2012. [Online]. Available: http://freecode.com/projects/gnumeric
[127] Kpresenter. Accessed December 5th, 2012. [Online]. Available: http://www.kde.org/applications/office/kpresenter
[128] L. Huston, R. Sukthankar, D. Hoiem, and J. Zhang, "Snapfind: Brute force interactive image retrieval," in *Image and Graphics, 2004. Proceedings. Third International Conference on*. IEEE, 2004, pp. 154–159.
[129] Sweet home 3d. Accessed December 5th, 2012. [Online]. Available: http://www.sweethome3d.com
[130] Travelling salesman problem. Accessed January 1st, 2013. [Online]. Available: http://www.tsp.gatech.edu
[131] N-queens problem. Accessed January 1st, 2013. [Online]. Available: http://www.math.utah.edu/~alfeld/queens/queens.html
[132] J. Ekanayake and G. Fox, "High performance parallel computing with clouds and cloud technologies," in *Cloud Computing*. Springer, 2010, pp. 20–38.
[133] U. Kremer, J. Hicks, and J. Rehg, "A compilation framework for power and energy management on mobile computers," in *Languages and Compilers for Parallel Computing*. Springer, 2003, pp. 115–131.
[134] Need for speed shift. Accessed May 15th, 2012. [Online]. Available: https://play.google.com/store/apps/details?id=com.eamobile.nfsshift_na_wf
[135] I. Kelényi and J. K. Nurminen, "Cloudtorrent-energy-efficient bittorrent content sharing for mobile devices via cloud services," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*. IEEE, 2010, pp. 1–2.
[136] D. Huang, X. Zhang, M. Kang, and J. Luo, "Mobicloud: building secure cloud framework for mobile computing and communication," in *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*. IEEE, 2010, pp. 27–34.
[137] W. Holden, "Mobile operator business models- challenges, opportunities & adaptive strategies 2011-2016," Juniper Research, Tech. Rep., 2011.
[138] Stratus. Accessed November 28th, 2012. [Online]. Available: http://research.microsoft.com/en-us/projects/stratus/
[139] Y. Qin, D. Huang, and X. Zhang, "Vehicloud: Cloud computing facilitating routing in vehicular networks," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*. IEEE, 2012, pp. 1438–1445.

Atta ur Rehman Khan is the C.E.O of DeeByte software solutions and a faculty member at COMSATS Institute of Information Technology (CIIT), Pakistan. He received his BS (Hons) and MS degree in computer science from CIIT in 2007 and 2010, respectively. Currently, he is pursuing his PhD from University of Malaya under the BrightSparks scholarship. His area of interest includes mobile cloud computing, sensor networks, VANETs and security.

Mazliza Othman is a senior lecturer at the Faculty of Computer Science and IT, University of Malaya. She received a BSc in Computer Science from Universiti Kebangsaan Malaysia, and a MSc and PhD degree from the University of London. She is the author of 'Principles of Mobile Computing and Communications' (Auerbach Publications, 2007). Her main interest is in pervasive computing and self-organizing network.

Sajjad Ahmad Madani is associate professor at COMSATS Insitute of Information Technology (CIIT), Abbottabad, Pakistan. He joined CIIT in August 2008 as Assistant Professor. Previous to that, he was working with the institute of computer technology (Vienna, Austria) from 2005 to 2008 as guest researcher where he did his PhD research. He received his MS degree in Computer Sciences from Lahore University of Management Sciences (LUMS). He received his BSc degree from UET Peshawar and was awarded gold medal for outstanding performance. His areas of interest include low power wireless sensor networks and application of industrial informatics to electrical energy networks. He has published more than 35 papers in international conferences and journals.

Samee U. Khan received a B.S. degree from Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi, Pakistan, and a Ph.D. from the University of Texas, Arlington, TX, USA. Currently, he is Assistant Professor of Electrical and Computer Engineering at the North Dakota State University, Fargo, ND, USA. Prof. Khan's research interests include cloud and big-data computing, social networking, and reliability. His work has appeared in over 200 publications. He is a Fellow of the Institution of Engineering and Technology (IET, formally IEE), and a Fellow of the British Computer Society (BCS).