Augmentation Techniques for Mobile Cloud Computing: A Taxonomy, Survey, and Future Directions

BOWEN ZHOU, The University of Melbourne, Australia
RAJKUMAR BUYYA, The University of Melbourne, Australia

Despite the rapid growth of hardware capacity and popularity in mobile devices, limited resources in battery and processing capacity still lack the ability to meet the increasing mobile users’ demands. Both conventional techniques and emerging approaches are brought together to fill this gap between the user demand and mobile device’s limited capabilities. Recent research has focused on enhancing the performance of mobile devices via augmentation techniques. Augmentation techniques for mobile cloud computing refer to the computing paradigms and solutions to outsource the mobile device computation and storage to more powerful computing resources in order to enhance mobile device’s computing capability and energy efficiency, e.g., code offloading. Adopting the augmentation techniques in the heterogeneous and intermittent mobile cloud computing environment creates new challenges for computation management, energy efficiency, and system reliability. In this paper, we aim to provide a comprehensive taxonomy and survey of the existing techniques and frameworks for mobile cloud augmentation regarding both computation and storage. Different from the existing taxonomies in this field, we focus on the techniques aspect, following the idea of realizing a complete mobile cloud computing system. The objective of this survey is to provide a guide on what available augmentation techniques can be adopted in mobile cloud computing systems as well as supporting mechanisms such as decision making and fault tolerance policies for realizing reliable mobile cloud services. We also present a discussion on the open challenges and future research directions in this field.

CCS Concepts: *General and reference → Surveys and overviews;* Computer systems organization → Cloud computing; *Software and its engineering → Cloud computing;* Human-centered computing → Mobile computing;

Additional Key Words and Phrases: Mobile cloud computing, mobile device augmentation technique

1. INTRODUCTION

Recent years have seen the exponential development of smart mobile devices. It has gained enormous popularity among mobile device users with more advanced mobile applications to enrich the user experience. However, compared to the growing demand for more enhanced user experience mobile applications from users, mobile devices are still in lack of adequate resources such as processing capability, storage and battery lifetime to provide such applications. The gap between the mobile hardware and users’ demand will hinder the mobile devices from providing experience-rich mobile services.

Due to the slow development of the battery technology, mobile resource augmentation has been considered as a critical approach to tackling the gap. Augmentation techniques for resource-constrained machines have been investigated since two decades ago. In the early 1990s, the ideas of remote execution and inter-process communication were introduced to help leverage the computing resources of computer clusters and manage message passing traffic [Gropp et al. 1994; Stevens 1990; Nelson 1981]. Remote execution aims to augment the computing capacity of resource-limited comput-
ers such as mobile devices and computers to run computation-intensive applications in a \textit{client-server} mode. However, utilizing the distributed resources can bring many challenges, including parameter marshaling, client and service binding, and the semantics of remote invocation.

Later on, with the development of the Internet and remote execution, a new service paradigm called Service-Oriented Architecture (SOA) is introduced. Mobile web services (MWS) [Pashtan 2005; Srirama et al. 2006] incorporates SOA with mobile devices to enable mobile device users to share existing software and services on other devices to augment its capability and conserve energy. Mobile devices in MWS can be either service clients or service providers.

Since remote execution and SOA rely on stable network connections, the performance of these systems can be unstable. Researchers study the possibility of utilizing the computing resources in the proximity (e.g., mobile devices nearby) with short-range wireless communications to provide an ad-hoc computing augmentation service. The earliest idea of the wireless ad-hoc network, “packet radio” network, was introduced by Defence Advanced Research Projects Agency (DARPA) in the early 1970s [Jubin and Tornow 1987]. Due to the development of the mobile device and 802.11/WiFi, the mobile ad-hoc wireless network (MANET) was then developed in the mid-1990s to provide a self-configuring mobile device network without the requirement of network infrastructures for collaborative mobile computing. The disadvantage of MANET is that the dynamic nature of network topology caused by the mobility of devices requires high adaptability of the network functions. Moreover, this type of mobile device augmentation can only provide limited resources within the resource pool built with other mobile devices. Later, pervasive computing [Satyanarayanan 2001] was introduced for a ubiquitous computing paradigm that enables computing to migrate from any device to any other devices via any type of network as the user moves. This paradigm essentially makes it possible to merge the computing resources of desktops, servers and mobile devices to provide continuing services.

Recently, cloud computing emerged as a promising computing paradigm in both industries and academia. Cloud computing aims to leverage virtualization technologies to provide computing resources such as computation, storage, and networks as a utility. The advantages such as on-demand self-service, broad accessibility, elastic scalability and pay-as-you-go services make cloud computing a potential computing resource for mobile device augmentation. The computing paradigm that leverages cloud computing resources to enhance the performance of resource-constrained mobile devices is called mobile cloud computing. In some works, it is also described as cyber-foraging [Lewis and Lago 2015], which refers to the same computing paradigm as mobile cloud computing. We use the term “mobile cloud computing” throughout this paper. A significant amount of research has been proposed in this area regarding the approaches for mobile cloud augmentation, resource provisioning and management [Cuervo et al. 2010; Kosta et al. 2012; Zhou et al. 2015b; Zhou et al. 2013; Soyata et al. 2012; Yang et al. 2015; Xiang et al. 2014]. These existing works focus on four main issues: the solutions of leveraging cloud resources, the efficiency of outsourcing the computation, the service availability and reliability, and the platforms for mobile cloud services.

As the hardware capabilities of mobile devices are developing rapidly nowadays, researchers have shown that the benefits of using the public cloud as augmentation resources for mobile cloud are decreasing [Srirama 2017], and the network conditions can be another performance bottleneck. On the other hand, utilizing a heterogeneous mobile cloud (HMC), namely a hybrid of mobile ad-hoc cloud, computers in proximity and public clouds, as augmentation resources can solve the issue. However, it also brings new challenges to the mobile cloud augmentation system.
Augmentation Techniques for Mobile Cloud Computing: A Taxonomy, Survey, and Future Directions

Many surveys have been given on the mobile cloud computing [Shiraz et al. 2013; Abolfazli et al. 2014; Sanaei et al. 2014; Li et al. 2015; Shuja et al. 2016]. Shiraz et al. [Shiraz et al. 2013] reviewed the existing distributed application processing frameworks for mobile devices to offload computational intensive applications to remote servers. It presented a detailed taxonomy and survey on the offloading frameworks including application partitioning, VM migration, partitioning granularity, migration objectives, etc. Abolfazli et al. [Abolfazli et al. 2014] presented a survey on the existing mobile code offloading frameworks based on the different types of cloud-based resources, which include distant immobile clouds, proximate immobile computing entities, proximate mobile computing entities, and hybrid cloud. However, its main focus is on the cloud-based augmentation frameworks comparison, without a detailed classification and discussion of the techniques that can be used for implementing a mobile cloud system. Therefore, the interoperability of the techniques mentioned may not fit in the HMC environment. Moreover, the survey only provided a brief discussion on the supporting techniques of mobile cloud augmentation such as decision making, context monitoring, and fault tolerance. Sanaei et al. [Sanaei et al. 2014] took a look into the heterogeneity in mobile cloud computing. They presented a taxonomy of heterogeneity of mobile device and cloud on four levels for both, including hardware, platform, features, and APIs. The existing approaches for handling the heterogeneity in mobile clouds were discussed. However, it does not include the discussion on realizing computation and storage augmentation of the hybrid mobile cloud. Li et al. [Li et al. 2015] focused on the survey of mobility-augmented cloud service provisioning and provided a taxonomy of cell network and cloud service provisioning mechanisms. Shuja et al. [Shuja et al. 2016] focused on multimedia application oriented survey that discussed application virtualization, dynamic binary translation techniques, and native code offloading based mobile cloud frameworks. These two surveys provided discussion only on specific problems, and do not meet the technical demands of the new heterogeneous mobile cloud systems. Different from the existing surveys, in this paper, we first introduce the concepts and issues of the HMC, and then we present a comprehensive taxonomy and survey of software techniques applied in the heterogeneous mobile cloud for both computation and storage augmentation.

The taxonomy covers the software augmentation related techniques for computation and storage and classifies with regards to implementation a heterogeneous mo-
A heterogeneous mobile cloud system in practice. Figure 1 illustrates a typical set of modules for the heterogeneous mobile cloud system framework, where the techniques presented in the taxonomy are needed. The HMC network consists of several HMC system powered devices. On each HMC device, the **Augmentation Engine** implements the main technique for computation and storage augmentation. The application submits offloading task requests to the **Decision Engine**, which makes offloading decisions based on its decision logic and the context data observed from the **Resource Monitors** and **Profilers**. The **Scheduler** further schedules the tasks upon receiving offloading decisions from **Decision Engine** to other HMC devices for remote execution. The **Fault Tolerant Manager** module supports the task execution in the event of failures. The **Communication** module deals with all the data-related transmission. The taxonomy presented in this paper is organized around modules of this system architecture.

In summary, the main contributions of this survey are:

- It introduces the main concepts and most challenging issues in realizing mobile cloud augmentation systems.
- It classifies the state-of-the-art techniques and works in a taxonomy with two parts regarding mobile computing augmentation and mobile storage augmentation respectively. Each type of techniques and works is discussed with advantages and disadvantages, as well as the suitable conditions for adopting that type of technique.
- The taxonomy is organized following the idea of necessary modules needed for realizing a practical mobile cloud service, including outsourcing techniques, supportive decision making mechanisms, resource management, and service reliability.
- It analyses the gaps still existing in mobile cloud augmentation systems and discusses future directions.

The rest of the paper is organized as follows. We explain the relevant terms and definitions, the motivation of mobile augmentation in the mobile cloud computing, and the key issues of mobile cloud augmentation in Section 2. Then we propose the taxonomy of computing and storage augmentation techniques and existing frameworks respectively in Section 3 and Section 4, followed by an analysis of the gaps and open challenges of mobile cloud augmentation in Section 5. Finally, we summarise the paper in Section 6.

2. BACKGROUND — MOTIVATION AND ISSUES

Different types of approaches have been applied for mobile computing and storage augmentation in mobile cloud computing. In this section, we explain the terms used in the previous works regarding the augmentation techniques. Furthermore, we discuss the motivation and key issues that have been targeted by the existing works.

2.1. Terms and Definitions

Due to the variety of techniques adopted in mobile cloud computing, we explain some of the important terms and conceptions that will be mentioned throughout the paper.

2.1.1. Cloud Computing. The cloud computing emerged in the recent years and gained popularity among both academia and industries. NIST gives the definition of cloud computing as “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models [Mell and Grance 2011].”
Fig. 2: Cloud Service Models

On-demand self-service: Customers can quickly provision the services automatically according to their demand (e.g., process capacity, network, and storage).

Broad network access: Customers can access the cloud service over the Internet by using heterogeneous client platforms such as mobile devices, laptops, and workstations.

Resource pooling: The infrastructure providers pool many resources from data centers and provide resource renting for clients in a multi-tenant service manner, with multiple physical and virtual resources dynamically assigned upon consumer demand.

Highly scalable: The service provider can easily scale up and down with the resources to meet user's demand automatically or manually when needed.

B. Service models

Generally, there are three types of service models provided by cloud service providers. The models can be summarized in a service stack, which is shown in Figure 2. From the bottom to the top, IaaS provides virtualized data centre infrastructure resources such as compute, storage and networking. Examples include Amazon EC2, Microsoft Azure, and Google Compute Engine; PaaS provides platforms and tools for application and other development, testing and deployment. Examples include Majsoft Aneka and Apache Stratos; SaaS represents the largest part of cloud services. It provides web-based applications where the backend services are managed on the cloud. Examples of SaaS applications include Facebook, Gmail, and Office 365.

C. Deployment models

Based on the difference of the administrative domain of the cloud, it can be further referred as the public cloud for general public users, private cloud for private groups and companies, and community cloud. The cloud computing service is adopted by a significant amount of research work as a potential approach for resource augmentation in mobile cloud computing.

2.1.2. Mobile cloud computing. In the beginning, mobile cloud computing was defined as a restricted computing paradigm that considers interactions between only mobile devices and public cloud services. Later on, due to the instability of mobile devices as well as wireless networks, more types of platforms and computing resources are introduced to mobile cloud computing to achieve seamless mobile cloud services. An overview of the heterogeneous mobile cloud service environment is shown in Figure 3. Based on the previous definitions and the new demands, we give the definition of mobile cloud computing as follows:
Fig. 3: An Overview of The Mobile Cloud Service Environment

Mobile cloud computing is a computing paradigm that enables the resource-constrained mobile devices to utilize heterogeneous computing resources (e.g., public clouds, private clouds, and MANETs) over multiple types of wireless networks (e.g., cellular network, WiFi, Bluetooth, and Femtocell) to provide mobile device users with a seamless, on-demand and scalable mobile service that has rich user experience.

The term “cloud” in mobile cloud computing refers to multiple types of computing resources. A brief classification of different clouds is as follows.

- **Infrastructure based cloud.** The infrastructure refers to public cloud services including IaaS, PaaS, and SaaS. The mobile device only outsources its computation and storage to the public cloud services via WiFi or cellular network.

- **Cloudlet based cloud.** Cloudlet refers to “trusted, resource-rich form-factor computer that is well-connected to the Internet and available for use by nearby mobile devices [Satyanarayanan et al. 2009]”. Cloudlets are deployed as middle layer servers between public cloud and mobile devices to reduce network latency.

- **Mobile device cloud.** It refers to a mobile wireless ad-hoc network (MANET), which consists of a set of mobile devices connected to each other via short-range wireless networks such as WiFi-direct and Bluetooth in dynamic topologies, with no support of networking infrastructure [Conti and Giordano 2014]. Mobile device cloud can further reduce the data transmission overhead and provide augmentation services in case the WLAN or public cloud services are unavailable.

2.2. Motivations

Mobile devices have already seen a significant improvement with more powerful and functional hardware such as multi-core processors, mobile graphic cards, and memory. Nevertheless, realizing mobile cloud computing can bring numerous benefits to both mobile device users and cloud service providers. From the user’s perspective, user
experience on mobile applications and functions of mobile devices can be improved by introducing mobile cloud computing. For cloud service providers, mobile clouds enable a large user community to use their services. Additionally, providers can apply machine learning mechanisms on the service data of mobile device users to further provide more customized cloud services.

2.2.1. Computing capability. Despite hardware improvements seen in mobile devices in recent years, the slow processing speed and low RAM still hinder mobile devices from providing experience rich applications to end users [Satyanarayanan 2011]. Since mobile devices have been involved as part of people’s daily activities such as social networking, office work, and gaming, they are expected to have PC approximate computing capacities. Mobile cloud computing provides the opportunities for developers to overcome this gap and enrich their applications by outsourcing computing-intensive tasks to cloud resources. Moreover, by migrating the computation to other computing resources, mobile devices are released to complete light tasks which eventually will improve the fluency of mobile devices.

2.2.2. Energy efficiency. One of the most urgent technical challenges for mobile devices is battery life. Currently, lithium-ion batteries used on mobile devices can only supply few hours of power for extensive computing. As more powerful processors, bigger displays and different types of sensors are installed on mobile devices, larger energy consumption is expected. However, the battery manufacturers are only able to increase the battery capacity by 5% annually [Robinson 2009]. Multiple efforts have been provided by phone manufacturers such as battery saving mode that dims displays or switches off wireless interfaces when the battery level is too low. Apparently, this type of approach neither solves the problem nor enhances user experiences. Therefore, mobile cloud augmentation can be a promising alternative solution that shifts the computation from mobile devices to other resources to save energy consumption on the battery.

2.2.3. Mobile storage. In the era of big data and mobile computing, many digital contents such as photos, videos, and large files have seen a drastic increase. Unlike the storage on PCs, mobile devices provide only limited flash storage space despite the demands of large and elastic storage from mobile applications and services. The elastic cloud storage resources can be a solution to the lack of mobile storage. The characteristics of cloud storage such as elasticity, on-demand, and low cost make it a potential extension to mobile devices. However, due to unstable wireless connections and mobile device mobility, the offline usability of mobile cloud storage augmentation is one of the challenges. Moreover, the migration of sensitive personal data from mobile applications to cloud storage raises the concern of data security and privacy.

2.2.4. Seamless and collaborative mobile application. With the advent of smart mobile devices, mobile users expect continuous mobile application experiences by using mobile cloud computing services. However, the wireless mediums that carry out data migrations are the performance bottleneck for mobile cloud services due to its instability and accessibility. In the event of network infrastructure failures, public cloud services may not be available, and mobile applications running its computation intensive tasks will be interrupted. In such a case, an ad-hoc network of mobile devices as a collaborative resource pool can be an alternate solution. Another example is the mobile social network in proximity (MSNP) [Chang et al. 2012], which enables mobile device users to interact with other users nearby via peer-to-peer wireless communications and complete tasks collaboratively.
2.3. Key Issues

The trends of utilizing heterogeneous computing resources and multiple types of wireless networks in mobile cloud augmentation bring many challenges. We present the major challenges related to techniques used in realizing mobile cloud augmentation systems.

2.3.1. Outsourcing computation. First, how to outsource the computation is one of the main challenges in the mobile cloud computing. Specifically, selecting the appropriate augmentation approach based on the characteristics of heterogeneous cloud resources is a non-trivial task. For instance, code offloading is one of the common approaches for outsourcing mobile tasks to cloud virtual machine (VM) based resources, while remote procedure calling is more suitable for SOA-based task integration. A few existing works such as MAUI [Cuervo et al. 2010], ThinkAir [Kosta et al. 2012], CloneCloud [Chun et al. 2011], MuSIC [Rahimi et al. 2013], Cloudlet [Satyanarayanan et al. 2009] have been proposed to solve the issue, but the efficient use of heterogeneous computing resources are yet to be studied more comprehensively.

2.3.2. Unstable wireless communications. Due to unique characteristics of wireless communication techniques, the network throughput, signal range, and connection stability can be downgraded by environmental changes like signal interference comparing to the wired networks. Another issue of wireless networks in the mobile cloud computing environment is the channel capacity. As the number of mobile device users grows, the service level agreement (SLA) regarding network performances is at risk of being violated. Therefore, handover strategies and seamless connection in case of a network SLA violation need to be studied when developing systems for mobile cloud augmentation.

2.3.3. Mobile device mobility. The mobile device management is vital to the performance of the mobile cloud augmentation system. On the one hand, the mobile device movement, together with the unstable wireless networks, can cause service failures in the mobile cloud augmentation systems frequently. On the other hand, the performance of the certain type of cloud resources like mobile device cloud (i.e., mobile ad-hoc networks) depends on routing protocols that are affected by the mobile device mobility. As a result, mobility management and fault tolerance need to be considered to provide a reliable mobile cloud service. The location-based device mobility prediction under different user context is another issue that needs to be investigated.

2.3.4. Context-aware augmentation decision making strategies. Outsourcing the computation to cloud resources is not always beneficial because the context of the mobile cloud environment changes rapidly. As we discussed above, the device mobility and unstable wireless networks are the main reasons that cause the change of the context. Moreover, the heterogeneity of the cloud resources and mobile applications can significantly affect the performance of the augmentation. For example, in computation outsourcing such as code offloading, transferring a significant amount of data for remote execution may consume more time and energy than running it on the mobile device itself. Instead, offloading this type of computation to a nearby mobile ad-hoc network may gain benefits from parallel execution and delay-free network transmission. Hence, it is necessary to develop augmentation decision making strategies to achieve context-awareness of the mobile cloud augmentation systems.

2.3.5. Data security. The computation and data outsourcing bring the concern of data security and privacy. The existing security challenges such as authentication, data integrity, privacy, and trust are inherited in mobile cloud computing due to the use of the remote cloud resources [Zissis and Lekkas 2012]. Ensuring the users’ privacy
when running their mobile applications on unknown computing resources is the main challenge of realizing mobile cloud augmentation systems in terms of security.

Many research works have been proposed with different techniques to tackle above-mentioned issues in heterogeneous mobile clouds. There have been both hardware augmentation technologies regarding battery [Ali et al. 2014], processor, etc. However, since our focus is on the development of heterogeneous mobile cloud systems for mobile cloud applications, this taxonomy covers the software augmentation technologies for HMC. A thematic taxonomy of the software mobile cloud augmentation techniques is depicted in Figure 4. It is divided into two main categories: computation augmentation and storage augmentation. For the computation augmentation, we present augmentation techniques related to task offloading for the Augmentation Engine in Figure 1, decision making techniques for devising offloading decisions for Decision Engine, with observed context monitoring data from Context Monitor and Profiler, and supporting techniques such as scheduling and load balancing, fault detection and recovery for the Scheduler and the Fault tolerance module. For the storage augmentation, we present the techniques related to storage offloading, data protection, and data interoperability. The detailed taxonomies of computation augmentation techniques and storage augmentation techniques are proposed in the following two sections respectively.

3. TAXONOMY OF COMPUTING CAPACITY AUGMENTATION

One benefit of mobile cloud computing paradigm is to bring the cloud resources to the device proximity to enhance the computing capability limited mobile devices. Multiple techniques can be leveraged to augment the computing capability of mobile devices. As shown in Figure 5, we will discuss the techniques in two aspects: the augmentation models and the augmentation architectures. Augmentation models, which include code offloading models and task delegation models, provide solutions of how the computation augmentation of mobile devices is realized by utilizing cloud resources. The augmentation architecture, which includes parallel execution and opportunistic mobile collaboration, describes the system architecture of the mobile cloud system for performing the computation augmentation. The two aspects work together to provide efficient mobile cloud augmentation services. A detailed discussion of these techniques is presented in the next four subsections.

3.1. Code Offloading

Offloading computation for remote execution is an idea that has been studied ever since the computer network was developed. It aims at migrating the computation intensive code from resource-limited machines to remote computing resources to accelerate the running process of the computation and reduce the energy consumption on
limited battery devices. Figure 6 shows a general concept of code offloading. In a general code offloading model, computation intensive codes of a mobile application are identified at first, and then it is evaluated by the decision making process based on the objective of the mobile cloud augmentation service (e.g., saving energy) to whether offload or not. Last, the code is offloaded to the remote computing resources by different types of available techniques.

The code offloading technique assumes the entire application computation originally happens on mobile devices. It enables mobile devices to migrate part of the computation from resource-limited mobile devices to resource-rich computing machines, which makes it flexible in terms of outsourcing. However, the disadvantage of code offloading is that it requires developers to identify and partition the part of the computation to offload, which is a non-trivial task and may impose unnecessary overhead for mobile devices. Various code offloading approaches have been proposed and applied based on various programming models of the mobile applications. The approaches can be classified into four categories: partitioned offloading, VM migration, mobile agents based offloading, and replication based offloading.

3.1.1. Partitioned offloading. For partitioned offloading methods, only the computation intensive part of mobile applications is identified and offloaded to the remote servers. The conventional client-server computing model is introduced to perform the partitioned offloading for HMC augmentation. Resource-limited mobile devices are considered as thin clients, while the remote computing resources that execute the offloaded codes are considered as resource-rich servers. The application is partitioned either statically before the runtime or dynamically at runtime. In the static partitioning, offloading partitions are decided and hard-programmed in the application by the developers. As a result, there is no overhead imposed by deciding partitions when the application is running. However, static partitioning requires the comprehensive program running knowledge for developers which is a non-trivial task. Also, the static partitions
make this offloading approach less adaptive to a dynamic computing environment like mobile clouds. Therefore, static partitioning is barely adopted in mobile cloud augmentation. To fix this issue, dynamic partitioning is introduced. Different from static partitioning, dynamic partitioning analyses the processes of an application at the runtime with the assistance of the offloading decision logic. Most commonly used dynamic partitioned offloading techniques are **Flow-based programming, .NET common language runtime programming, Java reflection**, and **distributed shared memory**.

**Flow-based Programming (FBP).** “FBP is a data flow programming paradigm that models applications as modulized processes connected with each other by pre-defined communication connections [Morrison 2010]”. These modules can be reconnected with each other to develop various types of applications without having to be modified. Therefore, each module can be dynamically decided whether to offload. To apply FBP in mobile cloud computing, Hung et al. [Hung et al. 2015] ported JavaFBP onto Android system. The Android applications are coded with APIs provided by JavaFBP in modules. However, using FBP as code offloading approach requires the installation on both mobile devices and offloaded machines, which hinders the scalability of the applications. To solve this problem, another approach using **Java Reflection** is proposed by Kosta et al. [Kosta et al. 2012].

**Java Reflection** is provided by Java that enables the program to inspect and manipulate the annotated classes, interfaces, fields and methods at runtime. Therefore, classes of an Android application can be decided dynamically at runtime whether to offload, and any machines with Java runtime can execute the offloaded partitions without requirements of additional installations. Kosta et al. [Kosta et al. 2012] proposed **ThinkAir** for mobile cloud code offloading on the method level using Java reflection. The annotated methods are evaluated by offloading decision logics dynamically and offloaded to cloud VMs running Android clones. In this way, the offloading services can be easily scaled up and down on cloud VMs. However, by using Java reflection, ThinkAir is only able to offload one method at a time and may cause lock-in issues.

In order to overcome this lock-in problem, Gordon et al. designed another solution named **COMET** [Gordon et al. 2012] for Android system using **distributed shared memory**. It is built on top of the Dalvik Virtual Machine [Ehringer 2010] on Android that leverages the underlying Java memory model and VM synchronization to form a distributed shared memory (DSM) for the code migration between machines. The offloading is implemented by synchronizing the information of heap, memory stacks, register states, and synthetic classes between VMs on cloud and mobile devices.
Besides Java on Android, other mobile platforms also provide similar functions that can be used for code offloading. .NET Common Language Runtime (CLR) “provides a managed coding platform featuring cross-language integration and exception handling and a simplified model for component interaction [Box and Pattison 2002]”. Cuervo et al. proposed a framework called MAUI that adopts .NET CLR on Windows phones to enable code offloading. Similar to Java Reflection, methods annotated by CLR tags are evaluated by MAUI’s decision making logic at runtime to decide whether to be offloaded. However, different from ThinkAir, MAUI requires users to develop a server version of the mobile application using CLR, which harms the application’s scalability in mobile clouds. Moreover, MAUI lacks consideration of multi mobile device offloading environment, and same lock-in issue as Java reflection remains.

The above-mentioned four types of techniques are the major approaches proposed. Compared with the static code offloading, although dynamic code offloading is more flexible and able to adapt to the different execution environment, it still requires modifications of mobile applications at the application developing stage. This drawback makes it non-trivial and tightly coupled with the underlying programming model of the mobile applications. With the development of new smartphone operating systems, many works later presented VM migration techniques for computation offloading.

3.1.2. VM migration. Live VM migration approach [Clark et al. 2005] for code offloading is based on the idea of moving the computation dynamically among the machines in the distributed system without interrupting the ongoing execution. It is fairly suitable for computation augmentation in mobile clouds since most mobile applications, as well as cloud services, are running on virtual machines. VMs can be migrated either partially or completely based on the requirement of the offloading. Applications are not required to be installed on the server side. However, the drawback of live VM migration is the overhead of transferring a VM image and its state via wireless networks can be costly and inefficient under the unstable network conditions. Hence, the focus of VM migration technique is improving energy efficiency considering the network conditions. Several previous works have proposed solutions based on live VM migration.

Satyanarayanan et al. proposed a VM migration based framework called Cloudlet1 [Satyanarayanan et al. 2009] to bring the computation from the mobile device to nearby form factor servers to overcome the long latency caused by transferring data to cloud via wireless networks. The term Cloudlet refers to “trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and available for use by nearby mobile devices.” Dynamic VM synthesis is developed to perform the transient partial VM migration service. The benefit of using Cloudlets is that it can be an alternative to the remote cloud resources to reduce the network bottleneck which often has significant effects on mobile cloud performances. However, the absence of monetary incentives of installing the Cloudlet-like machines is an issue to be further studied. Moreover, Cloudlets can only provide offloading service to stationary users. The continuous execution of the offloaded tasks would be an issue.

Chun et al. [Chun et al. 2011] took a different approach of using VM migration from Cloudlets. They proposed a code offloading framework CloneCloud that works with the application VM layer such as DalvikVM and Microsoft .NET. It deploys one or more clones of the mobile applications and data onto Cloud VMs and nearby servers and using process interception on the VM level to execute parts of the application on Cloud. The running states including data in the stack and heap are synchronized between the two processes on the mobile device and cloud VM. However, since the mobile clones have limitations on native data virtualizing, it does not have all the access to data on

1Source code can be found on https://github.com/cmusatyalab
the cloud side, which narrows the range of functions to be offloaded. This is also a vital issue for all VM migration based code offloading approaches.

3.1.3. Mobile agent based offloading. In order to overcome the data access limit of VM migration for code offloading, Mobile agent based offloading is introduced to mobile cloud computing. A mobile agent is a movable software that can be transferred from one mobile device to a network and roam among the computing nodes in the network [Chess et al. 1997]. When an application using mobile agents needs to request a service from the remote server like the cloud, it collects the application execution information and passes to the execution environment of the agent for offloading and remote execution. There are several advantages of using mobile agents in mobile cloud computing. First, the mobile agent uses asynchronous communication to interact with remote servers in the network, which is suitable for mobile devices since the network connections of mobile devices are unstable. Moreover, the device can still perform other lightweight computation while waiting for the results from remote execution. Second, mobile agents are robust to server failures, since all the information including executing environment, the process states, services required, etc. are transferable. Some mobile agent supported frameworks have been proposed.

Angin and Bhargava [Angin and Bhargava 2013] proposed JADE (Java Agent Development Environment)\(^2\) for mobile code offloading. In the event of task offloading, the method being offloaded is transferred into an agent-based process by the feature Behaviour in JADE on either class level or method level. Then it consults the cloud directory service to select one of the cloud hosts for the remote execution. JADE provides the offloading code with high movability and running adaptively. However, JADE does not consider the QoS of its cloud resources such as load balancing, application multi-tenancy and wireless channel energy efficiency for multi-users.

The multi-tenancy issue of mobile agent based offloading is investigated by Liu et al. [Liu et al. 2016]. They considered the energy efficiency of the wireless channels with multi-user scenario and solve the problem by using Lyapunov optimization framework to minimize the number of transmission time slots based on the queue backlog and channel states on each mobile device. However, this optimization approach only focuses on data transmission without considering local and cloud task execution, which may have a significant impact on the efficiency of the proposed algorithm.

3.1.4. Replication based offloading. Either partitioned offloading, VM migration or mobile agent based offloading requires applications to precisely analyse the benefits of whether to run portions of the application on mobile devices or remote machines. As a result, it usually benefits applications with large computation and small data size. In addition, the above three techniques do not benefit network-intensive applications. In order to solve this problem, Gordon et al. [Gordon et al. 2015] took a different approach by using replications. They proposed a framework called Tango for Android that attempted to reduce user-perceived application latency by switching execution and output display automatically between application and its replica on remote machines for the faster one. However, there are still some limitations of Tango. Only one replica can lead the execution instead of working simultaneously. This issue can affect the application running time as the leading replica may have no access to native resources such as sensors and user inputs. Also, the overhead of constantly switching is not presented in the work.

**Discussion.** The above-mentioned types of augmentation techniques adopted for code offloading aim to dynamically offload computation intensive code to cloud resources in order to conserve energy and speed up the execution. However, some code

\(^2\)Source code available at: http://jade.tilab.com/download/jade/
offloading approaches such as partitioned offloading require particular environment and comprehensive programming skills, which limits the usage on existing applications. Moreover, the lock-in issue of code offloading exists for approaches such as Java reflection and FBP. Also, the decision process of the offloading incurs additional overhead that will reduce the benefits of code offloading.

3.2. Service-oriented Task Delegation

Unlike the code offloading approach, service-oriented task delegation augments resource-limited mobile devices by utilizing existing services on remote servers with remote process invocation rather than migrating part of the application to the server for execution. OASIS has defined service-oriented architecture (SOA) as: “A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable pre-conditions and expectations [MacKenzie and Laskey 2006].”

A service is a self-contained unit that implements at least one reusable action such as searching a database or rendering a web page. The request to a service and its corresponding response are communicated via protocols such as XML and SOAP (Simple Object Access Protocol). The benefit of SOA for mobile cloud computing is that it provides a solution for multiple service providers and heterogeneous devices to incorporate their services into more comprehensive service combinations, where services are independent of any underlying vendors, products or techniques. The transparency of SOA-based services can reduce the additional issue and the execution lock-in issue of code offloading. SOA enables conventional applications to deliver their services to mobile devices via cloud resources with same user experiences on PCs, without the need of developing native mobile applications.

3.2.1. Web Service based Mobile Cloud. In the age of cloud computing, many legacy PC applications have been migrated to the cloud to provide the original functions in the form of web services. This SOA based service paradigm also enables mobile device users to use the legacy applications on their devices with the same PC-like user experiences via mobile cloud augmentation systems. The benefits of web service based mobile cloud systems are that there is no need to develop a native mobile application but only a thin client to relay the service requests to the cloud servers and render the returned results. In this way, the device is released to process other tasks and still able to obtain same, good quality of service. Many recent works have been proposed in the mobile cloud augmentation. Hani and Dichter [Hani and Dichter 2017] proposed a four-layer service-oriented architecture to provide an energy-efficient solution for web service based mobile cloud augmentation. The architecture ensures the security of the data handover and guarantees the quality-of-service. Rossi et al. [Rossi et al. 2017] designed a cloud-based service architecture to provide geolocated emergency services on mobile devices. Mobile users report the real-time condition reports about the emergency via mobile devices to the backend services. Crowdsourcing techniques are applied to evaluate the circumstances on the cloud. Guerrero-Contreras et al. (Guerrero-Contreras et al. 2017) studied the service availability in mobile cloud computing. They proposed a context-aware SOA based framework to improve the maintain service availability that can be downgraded by dynamic network changes in mobile cloud systems. The framework contains three services running on the cloud, including monitoring, context managing, and replica managing. As we can observe from these proposed works, one drawback of SOA based mobile cloud augmentation is service binding, which means mobile users can only choose certain provided services.
In the attempt to solve the operating binding issue, Flores et al. [Flores and Srirama 2014] proposed mobile cloud middleware MCM\(^3\) for processing the intensive hybrid cloud services regardless of the underlying mobile operating systems. The middleware provides a set of cloud service APIs available to the mobile device users to build the applications in its own language. MCM also fosters the integration and orchestration of mobile tasks delegated with minimal data transfer. However, the cloud services provided in MCM are limited, and the ability of users to add additional cloud services is yet to be delivered.

3.2.2. OSGi (Open Service Gateway Initiative). Another technique adopted in the SOA-based mobile cloud is OSGi\(^4\) [Alliance 2009], which is a Java module management system enabling applications to dynamically load and unload service bundles at runtime. The difference of OSGi from the C based RPC is that OSGi is object-oriented. The bundles can be invoked by users with a service interface, which is registered in the OSGi service registry with an implementation of the interface. Figure 7 depicts the OSGi system stack. Bundles are OSGi components built by developers. Execution Environment defines the available methods and classes for the user’s application. Modules define the way Bundles import methods and classes. Life Cycle provides APIs to install, start, stop, update, and uninstall bundles. Services dynamically bind the bundles with underlying Java objects. Services, life cycle, modules, and execution environment together define how the bundles work with underlying Java VMs.

Verbelen et al. [Verbelen et al. 2012] proposed an OSGi based middleware called AIOLOS\(^5\) for mobile augmentation on Android. The bundle feature is ported to Android. The mobile application is developed with OSGi bundles, service interfaces and remote services that implement the bundle services. Junior et al. [Junior et al. 2017] proposed a mobile offloading system (MOSys) to enable seamless offloading when users move between wireless networks. It is implemented based on OSGi, utilizes software-defined networking and remote caching to reduce the response time and provide a seamless handover. A few other works [Yang et al. 2016; Wang et al. 2016] have also presented OSGi based techniques to support the mobile cloud augmentation systems.

Discussion. SOA-based mobile cloud is fundamentally different from code offloading as there is no computation migration. Mobile devices in SOA work as thin clients that are only able to send service requests without any computation, while the main task computation is carried out in the form of services on the cloud. However, this

---

\(^3\)Source code available at: https://github.com/huberflores/MobileCloudMiddleware
\(^4\)More details at https://www.osgi.org/
\(^5\)The framework is available to download at: http://aiolos.intec.ugent.be/
limits the flexibility of mobile applications in terms of functions as it needs backend service to support.

3.3. Parallel Execution

More than one zettabyte of data is generated over the Internet nowadays [IBM 2016]. The needs of processing big data surpass the capacity of mobile devices. The big data applications such as mobile forensics applications, data streaming applications and augmented reality applications have been challenging conventional mobile cloud computing augmentation approaches. Although code offloading and SOA-based task delegation can enhance mobile devices with more computing capability, the overhead of migrating a large amount of data to the remote server is getting higher and higher. As a result, the benefits of outsourcing the computation are counteracted.

On the other hand, utilizing parallel execution to distribute computation can significantly reduce the time overhead of handling big data applications. In parallel computing, the computation and related data are divided into subproblems with a chunk of data, which are then processed simultaneously on multiple resources. Regarding data-parallel applications, sets of either homogeneous or heterogeneous task are processed in parallel, and the results are merged to generate the final result. The benefits of using parallel execution for mobile cloud computing augmentation depend on the types of applications particularly requiring parallel data processing. The frameworks such as **GPU computing**, **OpenCL** and **MapReduce** have been adopted to cope with big data applications in mobile cloud augmentation.

3.3.1. GPU computing. The thousands of cores on GPU enable applications to offload compute-intensive portions to the GPU while the remainder of the applications still runs on the CPU. Soyata et al. [Soyata et al. 2012] proposed a mobile cloud-based hybrid architecture (MOCHA) using GPU computing to enhance the performance of object recognition applications used on the battlefield. Typically, the battlefield handheld devices are connected via satellites that incur long latency. The Cloudlet in MOCHA is used to reduce the latency by letting mobile devices connect to Cloudlet in the vicinity via low latency links and then the Cloudlet processes either part or all the computation. GPUs are utilized to perform the massively parallel processing (e.g., object recognition), which provides massively parallel processing ability to applications by using CUDA\(^6\), which is a C-like programming platform and programming models designed for GPU parallel programming. However, GPU computing is only suitable for mobile applications that involve matrix-based computation, e.g., image processing due to the structure of cores on GPU. In addition, only intensive computation on small size data instead of streaming applications is preferred so that the speedup on execution would not be hindered by data transmission overhead. More importantly, GPU computing is only available with the certain brand of GPU, which causes vendor lock-in.

3.3.2. OpenCL. Different from GPU computing that only works with the particular type of applications, OpenCL [Bourd 2016] provides a more general parallel solution. It is an open-source platform for building parallel programs on cross-platforms that provide transparent C-compatible programming models for utilizing distributed processors regardless of the underlying architectures. Shih et al. [Shih et al. 2015] proposed an elastic computation framework for mobile clouds based on OpenCL frameworks. It federates computing and memory resources on wearable devices, mobile devices, nearby cloudlets and public cloud VMs as a shared resource pool for completing tasks on mobile devices. Eom et al. [Eom et al. 2013] studied the feasibility of applying HPC cluster architectures to the mobile cloud computing environment. They proposed an

OpenCL-based offloading framework that migrates OpenCL workloads from the mobile grid to clouds. OpenCL provides more portability and compatibility comparing to GPU computing since it works on both CPU and GPU.

3.3.3. **MapReduce.** Another approach having been adopted as augmentation technique in the mobile cloud computing is MapReduce. “MapReduce is a programming model and an associated implementation for processing and generating large data sets with a distributed algorithm on a cluster [Dean and Ghemawat 2008]”. Marinelli [Marinelli 2009] developed a mobile-cloud computing infrastructure called Hyrax based on Hadoop. Hyrax enables smartphone applications to run in distribution both regarding data and computation by adapting the Hadoop onto mobile devices within the Hyrax system. However, NameNode and JobTracker instances are not realized in Hyrax. In Hyrax, these must be run on a traditional machine. Duo and Kalogeraki [Dou et al. 2010] implemented MapReduce paradigm on Nokia smartphones using Python. The master node including a scheduler, an HTTP server, and an application repository is implemented on a server to map the jobs to other mobile devices as worker nodes. MapReduce is not widely used in mobile cloud computing since the processing capability of mobile devices is not enough for a complete MapReduce implementation.

**Discussion.** Different from the previous two augmentation techniques, parallel execution in mobile cloud aims to reduce the transmission overhead by breaking up big data into smaller chunks and process individually at distributed nodes at the same time in order to speed up the execution for resource-limited mobile devices. However, techniques like GPU computing have limitations on types of applications that can be run as well as vendor lock-in issues, which makes parallel execution restricted.

3.4. **Opportunistic Mobile Collaboration**

Mobile cloud augmentation techniques discussed above focus on utilizing remote computing resources via wireless networks like WiFi. However, most of these techniques rely on adequate wireless network bandwidth, which can be intermittent and cost-unfriendly. In addition, since mobile devices are usually on the move and may lose Internet connections in the area lacking signal strength, the communication between mobile devices and cloud services may be disrupted.

In recent years, the short-range wireless communication technologies on mobile devices such as Bluetooth, WiFi-direct, and Zigbee have been significantly improved regarding bandwidth and energy efficiency. With the development of both mobile device computing capacity and wireless networks, the mobile ad-hoc network (MANET) has become a potential mobile cloud infrastructure for augmentation. Many proposed works considered MANET as **mobile device cloud**, which is a network of mobile devices connected wirelessly in either centralized or distributed manner [Toh 2001]. The opportunistic mobile collaboration network provides a solution for mobile devices to dynamically utilize other peer devices in the vicinity to overcome the issue of unstable wireless network connections.

Many research works [Penner et al. 2014; Mao et al. 2016; Ravi and Peddoju 2014] have been carried out on applying MANET to enhance the mobile device performance. Penner et al. [Penner et al. 2014] proposed a collaborative computing platform, “Transient Clouds”, to enable mobile devices in the vicinity to share their own mobile application services to other devices in the manner of ad-hoc networks. The prototype was developed on Android using the built-in WiFi-direct feature. The tasks are stored in the auto-generated .dex file and later distributed to other devices by the decision engine. However, due to the limit of WiFi-direct, the ad-hoc network implemented is a one-hop network, which is not a complete implementation of the proposed framework and its feasibility needs further evaluation. Mao et al. [Mao et al. 2016] utilized
mobile edge device computing to offload computation in order to reduce network latency and congestion, and proposed an online Lyapunov optimization-based dynamic offloading algorithm to minimize execution cost by adopting DVFS. Zhou et al. [Zhou et al. 2015a] proposed a multi-tier mobile cloud service framework that includes mobile device cloud. The mobile device cloud is used for code offloading to obtain timely task execution and lower energy consumption for mobile cognitive applications like OCR, especially in the case of unavailable cloud services. The above-mentioned mobile device implementations all have only one type of wireless medium for communication, which leaves itself vulnerable to network disruptions, and also will not be scalable to devices with different wireless interfaces. Therefore, mobile device cloud that can utilize devices with multi-type wireless mediums needs to be investigated.

**Discussion.** The mobility and dynamic nature of MANET bring the reliability issue which needs the support of fault tolerant policies for a more stable performance. The approaches addressing the fault tolerance issues in mobile cloud augmentation systems are discussed in Section 3.8. In addition, the performance of MANET can be affected by network congestion as well as synchronizing the up-to-date information of the mobile devices.

### 3.5. Task offloading decision making

In order to provide efficient mobile task offloading services, decision making techniques are required to decide whether, when and where to offload. In section 3.5 and section 3.6, existing decision making techniques regarding task offloading, and task allocation and scheduling are discussed respectively. Figure 8 shows a detailed taxonomy of decision making techniques, including offloading decisions (whether to offloading), and task allocation and scheduling decisions (where to offload). For making offloading decisions, three main categories are discussed: stochastic process, analytic model, and resource monitoring and profiling. For making task scheduling decisions, the tech-
Augmentation Techniques for Mobile Cloud Computing: A Taxonomy, Survey, and Future Directions

Techniques are classified into four categories: heuristic, combinatorial optimization, meta-heuristic, and game theory.

In this section, the task offloading decision making techniques are classified into three categories: 1) stochastic process, which abstracts systems with random variables and statistically devises offloading decisions as systems evolve into stable states with certain probabilities; 2) historical data based analytic models, which aims to abstract systems with parametric models based on historical data and devise real-time offloading decisions unlike the stochastic processes; 3) resource monitoring and profiling, which provides real-time monitoring and profiling of the system and running applications, and utilizes the real-time data to devise offloading decisions without analysing on parametric models.

For both parametric model-based techniques and real-time monitoring data-based techniques, different factors of a mobile cloud system need to be captured in order to have valid analysis on the task offloading decisions. The factors can be classified as mobile application contents, contexts of the mobile cloud environment, characters of the hardware, and user preferences.

1) **Mobile application contents.** The contents refer to characters of applications including application type, code granularity, data size and type, user interaction requirements as well as the computation intensity. These application contents have important impacts on the results of the augmentation. For instance, applications with computing intensive jobs are more suitable for cloud servers if outsourced. Data streaming applications such as video streaming and Optical Character Recognition applications require short response delay while data analytic applications such as sentinel surveillance application in the medical field are more delay-tolerant. Hence, it is important to consider the differences in application contents when making augmentation decisions.

2) **Context of mobile cloud environment.** The mobile cloud environment is a resource sharing environment that consists of mobile devices, remote servers, and various wireless communication mediums. For example, wireless communication is one of the key components when outsourcing the computation, and the bandwidth and congestion will affect the time taken for transferring data, which will hinder the augmentation performance. Moreover, the mobility of mobile devices adds dynamics to the environment as the available resources are changing rapidly. Thus the decision making schemes should be agile for dynamic context changes.

3) **Hardware specifications.** The hardware refers to both mobile devices and remote servers, which have heterogeneous types of hardware such as CPU, memory, storage and communication modules. The challenge for decision making schemes is how to distribute the tasks among machines to either balance between time and energy saved by augmentation or fulfil user preferences based on the hardware as well as other factors discussed. Therefore, monitoring schemes are required to profile the hardware information to support the decision making schemes.

4) **User preferences.** Last but not the least, user preferences which are priority objectives for decision making schemes should be taken into consideration. User preferences can be very different depending on users’ circumstances. Some users want to use the augmentation service to save the battery of their mobile devices while other users wish to outsource the mobile tasks to get a shorter processing time regardless of how much battery it will consume.

3.5.1. **Stochastic process.** The stochastic process is a random process evolving with time [Cinlar 2013]. It aims to abstract the evolution of a system that changes based on random variations and predicts possible outcomes weighted by probability. Some well-known stochastic processes include Markov processes, Poisson process, and queueing theory. In mobile cloud augmentation systems, a stochastic process is applied to model
mobile applications as well as devices statistically, and evaluate the execution conditions to help devise optimal augmentation schedules.

Chen et al. [Chen et al. 2013] proposed a semi-markovian decision process (SMDP) based method to solve the problem of optimal offloading tasks dispatching and transmission for mobile cloud augmentation systems. Its objective is to obtain an optimal trade-off between computation time and energy consumption. The semi-Markov decision process based optimization is developed to abstract mobile device status and decide the probability of tasks to offload and at each stable state, which DVFS level of CPU and data rate of the transmitter to be set. The SMDP is solved with linear programming. One drawback is that many mobile devices have no access for users to change DVFS and wireless interface status.

Instead of focusing on a single device, Terefe et al. [Terefe et al. 2016] proposed an energy-efficient multi-site offloading policy for mobile clouds using Markov decision process (MDP). MDP is used to formulate application partitioning and scheduling to multi-clouds problem as a delay-constrained, least-cost shortest path problem with the goal to minimize energy consumption on wireless channels. Similarly, Wang et al. [Wang et al. 2017] applied Markov model in dynamic scheduling for mobile cloud health telemonitoring. As the mobile cloud infrastructure in health monitoring faces ever-changing clinical priorities and personal demands, the MDP based dynamic offloading scheduling algorithm proposed solves a joint optimization of processing latency, energy efficiency, and diagnostic accuracy. The stochastic process such as MDP can provide optimal solutions to multi-objective problems in reality. Nonetheless, the processing overhead of solving the problem and the sensitivity of the solution to new changes may pose a burden on the resource-limited mobile devices.

3.5.2. Historical Data Based Analytic models. Unlike stochastic process, this approach uses parametric models to abstract multiple attributes (parameters) of the mobile cloud augmentation environment, analyses the benefits of augmentation in terms of time and energy consumption based on historical data of the task execution. The attributes range from CPU speed of mobile devices as well as remote servers, network conditions (e.g., bandwidth, congestion, and latency), to the load of remote resources. However, the disadvantage of analytic models is that the types of attributes used in the models have to be consistent with the objective of the models. Too many attributes without proper assumptions and too few attributes without accurate abstraction may both downgrade the performance of the models. Many previous works applied this method to evaluate benefits of the mobile cloud computing augmentation.

Ali et al. [Ali et al. 2016] focused on modeling the power consumption of mobile devices to enhance the energy efficiency of mobile cloud augmentation systems. They presented detailed energy consumption models for CPU, display, wireless communication interfaces and memory on the mobile devices. For example, the energy model for CPU is based on the CPU frequency and utilization:

$$\text{Power}_{\text{cpu}} = \beta_{\text{freq}} \times U + \beta_{\text{base}}^{\text{freq}},$$

where $\beta_{\text{freq}}$ and $\beta_{\text{base}}^{\text{freq}}$ denote frequency dependent coefficients when CPU is in the busy and idle state respectively. $U$ is the CPU utilization. All the coefficients in the energy models are derived from a linear regression on the measurement results of smartphones. However, this regression approach is device dependent and may not be accurate when applied to other mobile devices.

Rahimi et al. [Rahimi et al. 2013] took into consideration device mobility when making augmentation decisions by modeling mobile applications as a location-time workflows. The mobile device is assumed to move in a partition of a 2D area. The mobility sensitive workflow models are derived by integrating the locations (i.e., coordinates) of
the mobile device and time duration of the device on that location into the workflow models. It is represented as follows:

$$W(u_k)_T^L ≜ (w(u_k)^{l_1}_{t_1}, w(u_k)^{l_2}_{t_2}, \ldots, w(u_k)^{l_n}_{t_n}),$$

where $u_k$ is the $k^{th}$ mobile user and $w(u_k)^{l_n}_{t_n}$ is the user workflow in location $l_n$ for time $t_n$. Such models can assist mobile cloud augmentation services to meet QoS requirements considering mobile device movements. However, the accuracy of the proposed models needs to be further studied and improved since the mobile user trajectory are obtained from some conventional mobility models such as Random Point Walk and Manhattan models [Camp et al. 2002].

Chen [Chen 2015] proposed a set of analytic models for processors and wireless network interfaces in terms of both computation and communication. For the computation models, the author proposed execution time models and energy consumption models which are both related to the computation size of tasks. The models are as follows:

$$T_n^l = \frac{D_n}{F_n^l}, \quad E_n^l = \nu_n D_n,$$

where $D_n$ is the computation amount, $\nu_n$ is the coefficient of the energy consuming rate per CPU cycle, and $F_n^l$ is the computation capability. Similarly, for the communication, the models are proposed based on the size of input data $B_n$, the network speed $R_n$, and wireless network power consumption rate $P_n$ of user $n$.

3.5.3. Resource monitoring and profiling. In order to overcome the issues of context adaptability in analytic model approach, resource monitoring and profiling are adopted in mobile cloud computing. This approach involves the implementation of multiple profilers and monitors on mobile devices to constantly monitor the context changes and the behaviour of the mobile device. Then the obtained data is put through the decision logic to evaluate the execution benefits and makes augmentation decisions based on the objective and user preferences. Many works have been proposed using this approach. We will discuss a few representative ones.

Benkhelifa et al. [Benkhelifa et al. 2016] proposed a genetic algorithm based resource augmentation for mobile device cloud to minimize energy consumption. A social cloud resource profiling and negotiating system are implemented, including a logger module which focuses on profiling the energy usage of applications on mobile devices under different circumstances such as using different wireless interfaces, executing tandemly with other applications, and executing at the different time of the day. However, the process of a large amount of gathered data against the storage limited mobile devices is not presented.

On the other hand, history-based resource profiling with compressed data storage model can solve the above issue. Kaya et al. [Kaya et al. 2016] implemented a set of profilers to monitor the application usage such as running time and CPU usage on mobile devices and cloud VMs. The history-based profiles are then constructed in call graph model that axes redundant information to reduce the storage. In case the application has never been executed before, the profiling mode would be invoked first and run several times with all possible use cases. However, this process puts unnecessary extra time for the application users, which would affect user experiences.

ThinkAir [Kosta et al. 2012] implements a set of profilers on Android that observe a more comprehensive range of system parameters to assist its Execution Controller for augmentation, including hardware profilers, application profilers, and network profilers. All the profilers are implemented using the system APIs provided by Android.

**Discussion.** In this section, we discussed three major types of offloading decision making techniques. The stochastic process discussed above provides a statistical ap-
approach of devising offloading decisions based on the states of the system and the transition probabilities obtained from different distributions. The accuracy of statistic distributions abstracting the mobile cloud systems can have a significant impact on the offloading decision. The analytic model approach has a similar component of system abstracts as stochastic processes, but instead, it devises offloading decisions based on historical execution data of the system rather than statistic distributions. Unlike these two approaches that are unable to the context change of system environment, the resource monitoring and profiling approach make offloading decisions based on real-time observations of task executions in mobile cloud systems. The main difference between the three approaches is the execution data that they make offloading decisions. Note that these three approaches can be used as combinations to provide more accurate results.

3.6. Task Allocation and Scheduling Decision Making

In a heterogeneous mobile cloud environment, each mobile device can outsource tasks via augmentation techniques to other resources using wireless networks. After the execution monitoring on the context parameters is completed, mobile tasks submitted from mobile devices need to be allocated to execute either in local or offload to other computing resources. Moreover, these tasks also need to be scheduled among the shared resources pool to achieve the objective of the system, such as optimal task completion time or minimum energy consumption. Therefore, task allocation and scheduling algorithms are required to distribute mobile tasks among the shared resources. Although there have been many task scheduling algorithms developed for distributed computing environments such as grid and cloud computing, the unique characteristics of heterogeneous mobile clouds such as intermittent wireless networks, unstable devices, and human-related behaviours make it difficult for the conventional scheduling algorithms to be applied. Hence, new algorithms and mechanisms are needed. Many task scheduling algorithms have been adapted to solve task allocation and scheduling problems in heterogeneous mobile clouds. They can be classified into four categories: heuristics, combinatorial optimization, metaheuristics, and game theory, based on their approach to solving the allocation and scheduling problem, as shown in Figure 8.

Heuristics are search algorithms that rank and select the solution subjects at each step of the searching process based on the most current information. They are often considered as a faster, less complex type of algorithms for solving optimization problems [Pearl 1984]. This is suitable for executing the algorithms on mobile devices that require lightweight processes. Greedy algorithms, listing heuristics, Min-Min algorithm are among many of the mostly used scheduling algorithms proposed for mobile cloud augmentation. Zhou et al. [Zhou et al. 2015b] adopted Min-Min algorithm to select the machine with least response time and energy consumption as the offloading destination for the independent task offloading. Li et al. [Li et al. 2015] proposed six online and batch scheduling based heuristics for scheduling independent tasks onto mobile nodes in mobile device cloud to reduce energy consumption. Trneberg et al. [Trneberg et al. 2017] proposed an iterative local search algorithm to find near-optimal solutions for application placement on mobile cloud resources. However, in exchange for the processing efficiency, heuristics are only able to provide local optimal solutions as they have no knowledge of the future events.

On the contrary, combinatorial optimization can devise global optimal solutions for the task scheduling optimization problems in mobile cloud augmentation. It searches the entire space of the feasible solution candidates and finds the best one as the optimal solutions. The computation efficiency of this method can be exhaustive if the search space is significantly large. Therefore, it usually deploys on a more powerful
 tier of computing resources in the heterogeneous mobile clouds. Methods such as (integ-er) linear programming, dynamic programming, and A* search algorithm have been applied in the mobile cloud augmentation. Gai et al. [Gai et al. 2016] presented a dynamic programming based algorithm to minimize the energy consumption of wireless communication in mobile cloud systems. Yang et al. [Yang et al. 2016] solved a joint optimization of task placement and load balancing with linear programming, and further proposed a greedy heuristic with low complexity. As we can observe, heuris-tics are possible alternatives when combinational optimization is infeasible to solve in real time. Another alternative is approximation algorithm, which aims to reduce the original problem to a similar, conventional problem, and is solved with approximation algorithms to the reduced problem. Some recent works have proposed solutions with this method [Sardellitti et al. 2015; Xu et al. 2016; Kao et al. 2017].

Another alternative is metaheuristic, which is a high-level algorithm to select and guide a heuristic to generate near-optimal solutions. It often refers to the nature-inspired algorithms such as genetic algorithm, ant colony algorithm, and simulated annealing. Metaheuristics usually adapt randomization and stochastic process in each of its evolutionary searches to reduce the search space and generate the solutions in a reasonable amount of time. Many recent works have been proposed to solve task scheduling problem in heterogeneous mobile cloud augmentation using metaheuris-tics such as genetic algorithm [Goudarzi et al. 2016; Benkhelifa et al. 2016; Gai et al. 2017], swarm optimization [Goudarzi et al. 2017; Qi et al. 2016], and ant colony optimization [Rashidi and Sharifian 2017; Wei et al. 2016].

Game theory based methods can also be leveraged to solve optimization problems, especially for mobile cloud augmentation where mobile users can be considered as players in task outsourcing games. It can abstract the heterogeneous mobile cloud in a more realistic approach in which the mobile device users can apply some strategies to maximize their interest. The game theoretic scheduling algorithm aims to find the optimal task augmentation schedules that can achieve a Nash equilibrium so that no strategy change from one player can change the results of optimal solutions. A few recent works have presented the game theory based algorithms for task offloading and scheduling in mobile cloud systems [Chen 2015; Chen et al. 2016; Tang et al. 2017].

Discussion. The task allocation and scheduling algorithms proposed in mobile cloud augmentation systems aim to provide optimal mobile task execution solutions to achieve the system objectives, such as optimal execution time and minimized mobile device energy consumption. As we discussed, the performance of these algorithms depends on the types of mobile applications, user preferences, SLAs (service level agreement), and the context of mobile cloud augmentation systems. Therefore, the algorithms need to be designed with the consideration of all the factors discussed.

3.7. Mobility Model for Mobile Device Cloud

In the following two sections, we discuss the mobility impact of mobile devices and fault tolerance techniques caused by the device’s mobility. These are the supporting techniques for the heterogeneous mobile cloud augmentation. A detailed taxonomy of the supporting techniques are presented in Figure 9. The mobility modeling is important for mobile cloud computing, especially for mobile device cloud which consists of an ad-hoc network of mobile devices. The decision making logics in a mobile cloud augmentation system can utilize the information obtained from the mobility models to estimate the availability and reliability of the mobile devices in the network and further make augmentation decisions. There are two types of mobility models commonly used: trace-based models and synthetic models. Trace-based models refer to the models built on real-world moving traces, which contains accurate information of the mobility
behaviors. However, the real-world traces require long period observation and a large number of participants. Therefore it is only useful case by case. On the other hand, synthetic models try to capture the mobility patterns without the assist of traces. The synthetic mobility models are categorized as entity mobility models and group mobility models.

3.7.1. Entity mobility models. Entity models aim to mimic the moving patterns of a single mobile node, without the interaction with other nodes in the proximity. We present four most commonly used entity models: Random Walk model, Random Waypoint model, random Gauss-Markov model, and Manhattan mobility model. A few available simulators such as ns2, ns3 can be used to simulate the node mobility.

The Random Walk model was firstly introduced by Karl Person [Pearson 1905] in 1905. It describes a walk path that consists of a series of random steps on a single or multiple dimension space. In the mobile cloud scenario, it is usually a 2-D space. At each stepping point, the mobile node chooses a random direction from \([0, 2\pi]\) and a random speed from a range set by the model, so that the patterns can be limited to a certain area. The Random Waypoint model is very similar to Random Walk model. The only difference is that Random Waypoint model adds a randomly chosen pause time to every stepping point. These two mobility models have been widely applied in mobile cloud computing [Taleb et al. 2016; Li and Li 2014; Mavromoustakis et al. 2015]. However, from the description, we can observe that Random Way model is a stateless random process that does not remember that information of previous steps. This property makes Random Walk generate unrealistic walking patterns since there may be sudden stops or sharp turns.

In order to solve the sudden stops and turns issue, the random Gauss-Markov model (RGM) can be applied in mobility modelling for mobile cloud augmentation. RGM moves mobile node in time intervals. At each time interval, the next location \(d_{next}\)
and speed $s_{next}$ are calculated based on its current location $d_{pre}$ and speed $s_{pre}$:

$$s_{next} = \alpha s_{pre} + (1 - \alpha) \bar{s} + \sqrt{(1 - \alpha^2)} s_{ran}$$

$$d_{next} = \alpha d_{pre} + (1 - \alpha) \bar{d} + \sqrt{(1 - \alpha^2)} d_{ran}$$

where $\alpha$ is the tuning parameter to vary the randomness. $\bar{s}, \bar{d}$ represent the mean values, and $s_{ran}, d_{ran}$ are two random variables from a Gaussian distribution. RGM avoids the sudden change issue by letting past states influence the future states. A few works have RGM implemented [Qiao et al. 2017; Antonescu et al. 2013; Miyake and Kami 2015].

For some special mobile devices like devices on vehicles, nodes move on an urban grid which includes only horizontal and vertical movements. Manhattan mobility model captures such movement patterns. In this model, each mobile node at each intersection is allowed to choose to go straight with 0.5 probability or to turn left or right with 0.25 probability. This model is implemented in [Rahimi et al. 2013; Jeong et al. 2016].

3.7.2. Group mobility models. Different from entity mobility models, group mobility models consider the influence of mobile device between each other that may affect the movement. In mobile cloud systems, especially mobile device cloud, nodes tend to move together such as a tourist group or a group of soldiers. Two most commonly applied models are Reference Point Group Mobility (RPGM) model and Reference Region Group Mobility (RRGM) model.

RPGM [Hong et al. 1999] is one of the most used group-based mobility models. Each group will have a central node as the leading node which follows an entity mobility model and sets the speed and direction for the entire group. The other mobile nodes in the network will be paired with a reference point that follows the leader point’s movement in same direction and speed. When the reference points move to the new location, its associated points will move to a random location within a circle of radius R around the reference point. A few RPGM based mobility models have been proposed for mobile cloud augmentation in [Huang et al. 2010; Ammari 2006]. However, the RPGM model should avoid using Random Walk based models that could generate sudden stops.

RRGM [Ng and Zhang 2005] further extends the RPGM to use a real-time determined sequence of regions to lead the group to some destination. The reference region is determined dynamically by user-defined node density and the size of the group. Members linked to the reference region will move to a random location in the region and then follows random waypoint model to wait until all members arrive in the region. The group will eventually move to the target destination following the reference regions. The advantage of RRGM is that it can mimic the real scenario where a large group can be divided into sub-groups and merge back together after some movements.

3.8. Fault Tolerance

As a computing environment with heterogeneous mobile devices, remote servers, and wireless network interfaces, maintaining a continuous service to avoid and recover from service interruptions caused by failures is one of the difficult challenges. Existing fault tolerance strategies proposed in distributed computing can be applied to mobile cloud augmentation environment with adaptations to its unique challenges such as device mobility and loss of wireless connections.

Mobile devices play the significant part of mobile cloud systems as it is not only the consumer of the systems but also can be resource providers (i.e., mobile device cloud). As resource providers, mobile devices can enter and leave the system environment unpredictably in case it moves out of the wireless network range or the network con-
connection is disrupted. Therefore, the faults considered in the context of mobile clouds are related to the mobility and availability of the mobile devices and remote servers as well as the stability of wireless network conditions. There are two main types of fault tolerance strategies: proactive fault tolerance and reactive fault tolerance.

3.8.1. Proactive fault tolerance. Proactive fault tolerance aims to prevent system faults by monitoring system through preventative measures to predict potential faults and prevent them from taking place. When a node failure is indicated, the fault tolerance mechanism preemptively migrates parts of the mobile application from the nodes about to fail. The basic form of proactive fault tolerance applies feedback-loop control over the distributed nodes in the system, i.e., mobile devices and cloud servers. The feedback loop consists of continuous node monitoring and reallocation of application partitions. However, proactive fault tolerance policies predict only before the task is dispatched, with no further action even if failure happens during task execution.

Hummel and Jelleschitz [Hummel and Jelleschitz 2007] proposed a proactive and reactive combined fault tolerance mechanism for the ad-hoc mobile grid. The proactive fault tolerance policy uses two mechanisms: redundant execution and super-peer access. Super peers refer to nodes that are constantly available and able to perform critical tasks. The unstable nodes use super-peer access to perform the execution and usually apply redundant execution on more than one node. Park et al. [Park et al. 2011] proposed a Markov process-based approach to analyse and predict resource states to improve the system’s resistance to fault problems related to the mobility of mobile devices. Ravi and Peddoju [Ravi and Peddoju 2014] presented a handoff strategy for the offload mobile tasks that proactively monitor the execution conditions of the mobile cloud augmentation system. The proactive failure evaluation adopts fuzzy logic based multi-criteria decision making algorithm. The handoff strategy is decided by monitoring energy consumption produced by using services from other resources and the remaining available time of mobile devices.

3.8.2. Reactive fault tolerance. On the other hand, reactive fault tolerance policies aim to reduce the effect of system faults that already happened. Mostly applied reactive fault tolerant mechanisms are checkpointing, and replication.

Checkpointing allows applications to restart at the most recent checkpoint when a failure is detected. Sonara [Chen et al. 2012] is a platform that aims to provide continuous mobile cloud augmentation service. Sonara’s execution engine enables a fault-tolerant distributed runtime that performs checkpointing based partial rollback recovery scheme. It adopted Chandy lamport’s snapshot protocol [Chandy and Lamport 1985] that carries out a global checkpointing throughout the system periodically.

Replication strategy is another well-known fault tolerance mechanism. In order to keep the service operate continuously after system faults occurred, multiple replicas of the task are distributed and run on different resources until the task is complete. However, replication will bring the challenge of adding redundancy into the system as well as synchronization problem. Choi et al. [Choi et al. 2014] proposed a fault tolerance scheduling algorithm for a content addressable network (CAN) based mobile cloud augmentation system. This is done by using cloud service replication. When the cloud server receives a request for cloud service, the server returns with two or more proper resources from other mobile devices that can meet the QoS and operate the service on all the resources.

A summary of the existing implemented frameworks in the mobile cloud computation augmentation techniques is presented in Table I.
4. TAXONOMY OF MOBILE STORAGE AUGMENTATION

Mobile devices are equipped with limited storage that cannot match mobile application's growing needs of larger data storage consumption. On the other hand, public cloud also provides the Data-as-a-Service (DaaS). DaaS enables users to offload a large amount of data to cloud storage services that mobile device storage cannot provide. For example, Amazon web service provides Simple Storage Service for object storage and charges based on the usage. Therefore, data storage augmentation by storing mobile data on cloud servers is of interest for mobile cloud augmentation systems. The sensitive data transmission such as location coordinates and healthcare information via wireless communication mediums and data storage on other computing resources bring security challenges into mobile cloud computing. We present a taxonomy of mobile storage augmentation in Figure 10.

4.1. Mobile Storage Offloading

Despite that mobile devices have been improved regarding hardware, the lack of substantial storage and computing capacity are still hindering mobile devices from better user experiences. Cloud has become a primary resource for enhancing mobile devices in terms of computation as well as storage. Many approaches and techniques have been adopted in mobile cloud storage augmentation to solve the problems such as storage augmentation, data distribution, data access, and data synchronization.

Most commonly used mobile storage augmentation solutions involve public cloud storage services. In order to extend mobile storage, the solutions propose middlewares between cloud services and mobile devices to provide data offloading and data management functions. Zheng et al. [Zheng et al. 2010] presented a novel cloud storage augmentation framework called SmartBox. It introduces a concept called “shadow storage” services to extend the storage on mobile devices. The shadow storage service will automatically back up the data stored on mobile devices to the cloud storage when the device is connected to SmartBox, and share data between different mobile devices. Hung et al. [Hung et al. 2015] proposed a mobile storage augmentation system based on Trusted Architecture for Securely Shared Service (TAS3) [Kirkham et al. 2011]. It is a secure network for user-centric storage by using a rule-based policy framework to let service users store and process their private data in distributed applications on both mobile devices and cloud servers. However, mobile storage augmentation using public cloud services heavily relies on wireless networks, which makes it a disadvantage in case there are wireless network outages.
## Table I: Comparison of Mobile Cloud Computation Augmentation Frameworks

<table>
<thead>
<tr>
<th>Framework</th>
<th>Cloud Resource Type</th>
<th>Code Offloading</th>
<th>SOA Task Delegation</th>
<th>Parallel Execution</th>
<th>Opportunistic Collaboration</th>
<th>Context-aware Execution Monitoring</th>
<th>Task Allocation &amp; Scheduling</th>
<th>Mobility Management</th>
<th>Fault Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileFBP</td>
<td>Nearby servers</td>
<td>Flow-based programming</td>
<td>—</td>
<td>—</td>
<td>Resource profiling</td>
<td>Single user, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>MAUI</td>
<td>Nearby servers</td>
<td>Not common language runtime</td>
<td>—</td>
<td>—</td>
<td>Resource profiling</td>
<td>Single user, minimum makespan and energy, ILP</td>
<td>—</td>
<td>—</td>
<td>Checkpoint</td>
</tr>
<tr>
<td>ThinkAir</td>
<td>Public clouds</td>
<td>Java reflection</td>
<td>—</td>
<td>Resource profiling</td>
<td>Thread execution profiling</td>
<td>Single user, heuristic</td>
<td>Single user, heuristic</td>
<td>—</td>
<td>Memory error detection, Checkpoint</td>
</tr>
<tr>
<td>COMET</td>
<td>Public clouds</td>
<td>Distributed shared memory</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Cloudlet</td>
<td>Nearby servers</td>
<td>Live VM migration</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Single user, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Clonecloud</td>
<td>Public clouds</td>
<td>Live VM migration</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Single user, minimum makespan and energy, ILP</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>JADE</td>
<td>Public clouds</td>
<td>Mobile agent(Java)</td>
<td>—</td>
<td>—</td>
<td>Analytic models</td>
<td>Single user, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Scavenger</td>
<td>Nearby servers</td>
<td>Mobile agent(Python)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Multi users, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Spectra</td>
<td>Nearby servers</td>
<td>Remote procedure call(RPC)</td>
<td>—</td>
<td>Resource profiling</td>
<td>—</td>
<td>Single user, heuristic</td>
<td>—</td>
<td>Proactive prediction on usage demand</td>
<td></td>
</tr>
<tr>
<td>Cuckoo</td>
<td>Nearby servers</td>
<td>Android interface defined language</td>
<td>—</td>
<td>—</td>
<td>Not required</td>
<td>Single user, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SAMI</td>
<td>Public clouds</td>
<td>SOA-based middleware</td>
<td>—</td>
<td>Traffic profiling</td>
<td>Not required</td>
<td>Multi users, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>MCM</td>
<td>Public clouds</td>
<td>SOA-based middleware</td>
<td>—</td>
<td>—</td>
<td>Not required</td>
<td>Multi users, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>AIOLOS</td>
<td>Public clouds</td>
<td>OSG</td>
<td>—</td>
<td>Not required</td>
<td>Multi users, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>AlfredO</td>
<td>Mobile device cloud</td>
<td>OSG</td>
<td>—</td>
<td>Not required</td>
<td>Multi users, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>MOCHA</td>
<td>Mobile device cloud and public cloud</td>
<td>GPU computing with CUDA</td>
<td>—</td>
<td>Analytic models</td>
<td>Multi users, heuristic</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Hyrax</td>
<td>Mobile device cloud</td>
<td>MapReduce</td>
<td>—</td>
<td>—</td>
<td>Analytic models</td>
<td>Multi users, heuristic</td>
<td>—</td>
<td>—</td>
<td>Redundant execution</td>
</tr>
<tr>
<td>[Shih et al. 2015]</td>
<td>Mobile device cloud</td>
<td>OpenCL</td>
<td>—</td>
<td>—</td>
<td>Resource profiling and discovery</td>
<td>Single user, heuristic</td>
<td>—</td>
<td>Monitoring error code implemented in each method</td>
<td></td>
</tr>
<tr>
<td>mCloud</td>
<td>Mobile device cloud</td>
<td>Live VM migration</td>
<td>—</td>
<td>—</td>
<td>Ad-hoc network</td>
<td>Single user, heuristic</td>
<td>—</td>
<td>Heartbeat, checkpoint</td>
<td></td>
</tr>
<tr>
<td>Transient Clouds</td>
<td>Mobile device cloud, Cloudlet, public cloud</td>
<td>—</td>
<td>—</td>
<td>Ad-hoc network and discovery</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Handoff scheme</td>
<td>Mobile device cloud</td>
<td>SOA-based middleware</td>
<td>—</td>
<td>—</td>
<td>Ad-hoc network</td>
<td>Resource profiling and discovery</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FollowMe Cloud</td>
<td>Mobile device cloud</td>
<td>SOA-based middleware</td>
<td>—</td>
<td>—</td>
<td>Ad-hoc network, edge computing</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MobiCloud</td>
<td>Mobile device cloud</td>
<td>SOA-based middleware</td>
<td>—</td>
<td>—</td>
<td>Resource profiling</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
In order to avoid such wireless network bottleneck, some approaches applied mobile device cloud as the storage augmentation resources. Phoenix [Panta et al. 2013] is a protocol designed to make opportunistic use of mobile devices in the MANET to provide a short-term storage service to clients in the proximity of each other. The distributed and asynchronous protocol breaks content into blocks and stores copies of blocks on multiple mobile devices so that it can ensure some degree of storage redundancy despite hardware failures, device mobility, and wireless communication failures. Additionally, Phoenix implements an advertisement model for maintaining and managing the blocks among the distributed mobile device storage. Similarly, Chen et al. [Chen et al. 2015] proposed a fault-tolerant data storage management algorithm for mobile device cloud that only contains mobile devices. The algorithm introduces a “k-out-of-n” strategy, which is a well-studied reliability control strategy [Coit and Liu 2000]. The strategy distributes multiple copies among the number of n mobile devices and ensures that the system operates correctly as long task copies are available.

Different from using mobile device cloud to solve the network bottleneck, Cui et al. investigated the wireless network bandwidth and data sync traffic directly. They proposed a system called QuickSync to improve the synchronization inefficiency problem in mobile cloud storage augmentation services. Three key components are implemented, namely Network-aware Chunker (NC), Redundancy Eliminator (RE), and Batched Syncer (BS) to reduce the redundant data generated by sync traffic in the network.

4.2. Data Protection

For most mobile applications such as healthcare applications and OCR applications that get benefits from mobile cloud augmentation services, the data in transmission often contains sensitive and private information, and therefore needs to be protected. We discuss three major related issues, namely data security, accessibility, and authentication.

4.2.1. Data security. There are three important attributes of data security, which are confidentiality, integrity, and availability.

**Confidentiality** guarantees that the sensitive data is only exposed to users with proper authority. One of the commonly used approaches for confidentiality is data encryption. By encrypting the data with private information (e.g., public/private key pair), it can only the authenticated users have access to the data. Examples include SSL/TLS and HTTPS which are communication protocols for secure information exchange.

**Data integrity** maintains the consistency and accuracy of the data. The most widely applied methods for protecting data integrity include data hashing technologies such as MD5 and SHA that capture the signature of the original data with certain hashing methods and compare it with the received data.

**Data availability** refers to the property that authorized entities can exclusively access the information on demand. A common solution for improving data availability is doing regular offline data backups, which is used to restore the data and services when the original data is damaged or under attack like the DDoS attack. Many works have been proposed for the mobile storage augmentation to ensure the above-mentioned three attributes of data security. Common topics include encryption, access control, authentication, data synchronization, and privacy.

Zhou and Huang [Zhou and Huang 2013] present a new Privacy Preserving Cipher Policy Attribute-Based Encryption (PP-CP-ABE) to ensure the confidentiality of the data. The encryption algorithm is based on CP-ABE [Bethencourt et al. 2007], which is used to simplify the process of key pairs generating and access control. However,
CP-ABE requires intensive computation that is not suitable for mobile devices. PP-CP-ABE provides a solution to this problem by outsourcing encryption and decryption operations to cloud with privacy preservation by performing the last step of decryption at the decryptors.

Yuan and Yu [Yuan and Yu 2014] investigated the data integrity checking techniques for cloud data sharing with multi-user data modification. They proposed integrity checking scheme with constant computational cost by using polynomial-based authentication tags which allow aggregation of tags of different data segments. Therefore, the scheme can tag files in batches from different users on the cloud and ensure a constant performance with scalability. Wang et al. [Wang et al. 2013] proposed a similar approach but without the need of entire data file for integrity checking. This is enabled by utilizing re-signatures for file blocks on a public verifier.

Li et al. [Li et al. 2013] adapted Chase and Chow’s attribute-based encryption (ABE) scheme [Chase and Chow 2009] into mobile cloud computing with substantial communication and computation overhead reduction to provide a low complexity multi-authority ABE scheme (MA-ABE) mobile devices sharing their storage. The overhead reduction is made by introducing a cloud server based semi-trusted-authority, where the computation of keys for encryption and decryption are taken places.

Different from the above approaches, Khan et al. [Khan et al. 2014] consider the limited resource on mobile devices when adopting data encryption for mobile cloud computing. They proposed an increment-based proxy re-encryption scheme that improves file modification operations. A trusted entity was introduced so that the re-encryption services are offloaded on it instead of performing on the mobile device itself.

4.2.2. Authentication and access control. In mobile cloud storage augmentation systems, mobile users usually share files among multiple computing resources as well as other mobile users. To support the data protection requirements, mechanisms for access control and authentication need to be provided to ensure only the verified user groups have access to certain files.

Zhou and Huang [Zhou and Huang 2013] proposed a set of access control schemes called Attribute Based Data Storage (ABDS) for energy-efficient and secure storage augmentation services in mobile cloud computing as well. The ABDS access control is managed by an access policy tree that consists of leaf nodes and branch nodes. The leaf nodes represent parameters that carry the information of the access request, and each branch node is a logical gate, such as “AND” and “OR”. The mobile users requesting the MSA services will be determined by each access policy tree defined for different user groups.

Lomotey and Deters et al. [Lomotey and Deters 2013] designed a framework called ALILI to solve the group file sharing problems in mobile cloud systems. ALILI aims at ensuring data synchronization and user authentication among the mobile device users as well as cloud service platforms. The authentication is based on OAuth 2.0 mechanism to enable the user with secure tokens and basic information retrieved from social media credentials servers. It also makes the authentication process easier to map the user shared files in a shared storage.

Wang et al. [Wang et al. 2015] proposed a key distribution mechanism for mobile devices in Internet of Things, considering real-time data collection and monitoring. The proposed mechanism secures real-time key distribution in batch for the parallel mobile services while keeping the communication cost consistent with the number of mobile clients.
4.3. Data Interoperability

The heterogeneity of mobile cloud augmentation systems brings the problem of data interoperability. Since mobile devices may have different operating systems and hardware settings, the application data, and APIs for communicating with each other may vary in data formats. Issues such as data interoperability between various service APIs on cloud and mobile devices, data portability among different types of data warehouse facilities, and data migration from mobile devices to cloud services or across different cloud service vendors need to be studied.

One of the solutions is applying standardized service frameworks and message exchanging techniques such as SOA, REST, XML and JSON to mobile augmentation systems. Abolfazli et al. [Abolfazli et al. 2012] proposed a SOA-based mobile device cloud called MOMCC. Since SOA defines standard service APIs for mobile devices and cloud services and services apply SOAP to exchange messages, mobile applications only need a small amount of modification to provide services across heterogeneous platforms.

Mobile social networks (MSN) are social networking where mobile device users having common interests connect and interact with each other through their mobile devices. In MSNs, the heterogeneity of software platforms on mobile devices and intrinsic user data and content raise the need to develop uniformed mobile application platforms. Toninelli et al. [Toninelli et al. 2011] proposed a middleware called Yarta for mobile social systems. To achieve this, a representative model based on the Resource Description Framework 7 is presented. The mobile social applications developed with the representative model can share and reuse their respective data interoperably.

Doukas et al. [Doukas et al. 2010] presented a mobile healthcare application framework for Android OS that provides trusted healthcare information online storage, retrieval, and update using cloud services. The data management of healthcare data in mobile cloud augmentation systems involves problems such as data privacy and interoperability. They presented a data management system called HealthCloud that implements a series of REST APIs for utilizing the storage services on the cloud. To ensure the data security, all the transferred data are compressed and sent via SSL enabled links.

5. GAP ANALYSIS AND FUTURE DIRECTIONS

Although many aspects of mobile cloud augmentation systems have been studied by previous works, there are still technique gaps between the proposed solutions and comprehensive mobile cloud services. In this section, we identify the technique gaps and related challenges, and present future directions in mobile cloud augmentation ecosystems.

5.1. Heterogeneous Mobile Cloud Service

As mobile devices become prevalent in people’s life, various types of smart mobile devices such as wearable devices, smart home appliance, mobile phones, and tablets have significantly affected the ecosystem of mobile cloud augmentation. Different types of mobile devices have different capabilities. For example, wearable devices are regarded as sensors that provide user-related information without further processing and analysis, while smartphones have the computing capacity to execute more complicated mobile tasks. Therefore, a multi-tier mobile augmentation service (MAS) framework that utilizes resources from all types of available mobile devices, as well as cloud resources, is of interest. The framework needs to be general enough to adapt mobile devices and machines that run different operating systems. Moreover, the heterogeneity of mobile

7https://www.w3.org/TR/rdf-primer/
devices and servers in terms of system and data format brings the problem of data interoperability that needs to be studied.

5.2. Context-awareness
Within MAS systems, the context information such as user mobility, social information among users, network conditions, and device information can provide additional assistance for decision modules to devise more comprehensive mobile augmentation solutions based on different objectives. The multi-tier heterogeneous MAS frameworks make it possible to enable MAS providers to process the context of MAS to improve the quality of their services and bring the opportunities for mobile cloud social-aware applications. However, the rapidly changing execution environment hinders the efficiency of continuous context monitoring and analysis that produce considerable overhead on mobile devices. Therefore, designing resource and energy efficient task allocation and scheduling mechanisms for multi-tier MAS systems is necessary.

5.3. Quality of Service Management
The Quality of Service (QoS) in mobile augmentation services refers to the criteria such as service response time (delay), constant wireless communication, availability and scalability of services, the fair use of services, and mobility management. For mobile device based systems, the fundamental problem for improving QoS is mobility management. Hence, an efficient mobility management scheme to estimate mobile device’s available time within the MAS system is in crucial need. With the support of mobility management, we can further improve the service response time by utilizing computing resources such as other mobile devices and Cloudlets in vicinity based on the mobile task requirements. On the other hand, it is challenging to design a service model that can manage a large number of mobile clients and wireless communication system while ensuring the availability of services. Therefore, efficient and continuous provisioning of resources and services in mobile augmentation service systems is a research perspective that needs investigation.

5.4. Reliability of Mobile Cloud Augmentation Systems
The heterogeneous mobile cloud system is a dynamic computing environment that the availability of computing resources and network conditions may change constantly. To provide mobile device users with reliable and seamless mobile cloud services, fault tolerance mechanisms are required to reduce the impact of service outages. However, only few fault tolerance schemes are proposed targeting the specific type of mobile cloud environment (i.e., mobile ad-hoc network). Therefore, new reliability mechanisms adopting both proactive and reactive fault tolerance schemes need to be investigated for the nested computing environment of mobile clouds.

Containers have drawn increasing attention in public cloud services and businesses. It provides a light virtualization approach as an alternative to VMs. Containers can run on multiple operating systems and have high portability. Future research can investigate the possibility of engaging container technology into mobile cloud computing in order to improve the reliability and scalability of mobile cloud services.

5.5. Security and Privacy
Due to the data transmission between mobile devices and other computing resources like cloud and mobile devices in the vicinity, data safety, and privacy are important concerns. Despite there is an enormous amount of research have been proposed for security issues in the cloud, it is still one of the major gaps in mobile cloud-based systems. First, the security and privacy mechanism needs to be lightweight to reduce the overhead on mobile devices. Second, due to the large amount of data transmission
over wireless networks, data protection for integrity and confidentiality in wireless networks need to be considered. Last but not least, a trustworthy distributed computing model is expected to cope with the computation taken place on the remote server and mobile devices and prevent unauthorized access as well as potential data leakage.

5.6. Incentives for Service Users
To build a fair use mobile augmentation service, an incentive mechanism is required to convince mobile device users to opt-in to the system. The incentive mechanism needs to provide proper motivation for the users to share their resources from mobile devices to the shared resource pool within the mobile cloud augmentation system. However, finding efficient motivation mechanisms has been found difficult for individual concerns such as information security and privacy, the limit battery lifetime on their mobile devices, and different demands on the mobile augmentation service. Therefore, designing an appropriate incentive mechanism for mobile cloud augmentation systems is of interest for future research.

5.7. Virtual Reality, Augmented Reality and Artificial Intelligence on Mobile Devices
The larger display and more powerful hardware on mobile devices make it widely popular to build augmented reality (AR) and virtual reality (VR) mobile applications. These types of mobile applications require constant camera data streaming of the captured frames on cameras and always-on display. In addition, artificial intelligence (AI) technologies are used to enable devices to make cognitive recognition like human beings to discover useful information from the captured data. Recent research has seen the development in this field. FlashBack [Boos et al. 2016] is a system proposed for VR head-mounted devices to pre-compute and cache all possible encountering images to reduce the computational burden on the device GPU. Furion [Lai et al. 2017] is a mobile VR framework that enables QoS focused application developing on mobile devices using cloud offloading. MobileDeepPill [Zeng et al. 2017] is a mobile AR system for smartphones to help identify unknown prescription pills captured by the phone’s camera by using a proposed deep learning image recognition algorithm. The constant data streaming and high computational requirement of AI from these types of applications yield challenges for mobile devices regarding energy efficiency, data streaming management, pre-stored data management for reuse, and network throughput optimization. Designing efficient mobile cloud offloading system for these applications can provide possible solutions.

6. SUMMARY
With the prevalent mobile devices and the development of hardware capability, especially smartphones, mobile device users demand a more comprehensive and advanced user experience from mobile applications that mobile devices are not able to provide in terms of computing capacity and storage. Augmenting computing capabilities and storage of mobile devices is a promising solution to fulfill the gaps. The ultimate goal of mobile augmentation solutions is to provide scalable, continuous, and secure PC-like services for mobile users regardless of the underlying limited mobile device resources.

This paper presents a comprehensive taxonomy and survey on the augmentation techniques applied in mobile clouds. Firstly, the terms and definitions used throughout the survey are explained, including cloud computing and mobile cloud computing concepts. Then, the current issues and challenges studied in the literature are presented. Secondly, the existing mobile cloud augmentation techniques for both computation and storage are discussed in two detailed taxonomies. The taxonomy of computing capacity augmentation discussed the approaches and techniques that have been applied in mobile cloud augmentation to merge the hybrid cloud resources into a shared
resource pool for mobile devices to provide reliable and energy-efficient computation outsourcing mobile-cloud-as-a-service. The advantages and disadvantages of each type of technique are compared, and future directions to further tackle the disadvantages are discussed at the end of each section. Regarding mobile storage augmentation, the taxonomy discussed the data-oriented architecture for storing data on distant clouds as well as mobile device cloud. Moreover, it covered the most critical issues in mobile cloud augmentation systems, namely data protection and data interoperability. As the mobile computing technologies evolve rapidly, new applications and techniques are released to mobile users, which can create technical gaps between current mobile cloud services and the new user demands. The survey analysed seven major technical gaps can be further studied in future, including service heterogeneity to incorporate new mobile devices such as wearable devices and IoT devices, service context-awareness and QoS management to agilely adapt the change of device context, reliability and security management of mobile cloud systems as modern smartphone operating systems such as Android and iOs can be vulnerable, incentive mechanisms, and AI empowered VR/AR mobile applications using mobile cloud services.

ACKNOWLEDGMENTS

We thank Rodrigo N. Calheiros, Amir Vahid Dastjerdi and Satish Narayana Srirama for their comments on improving the paper.

REFERENCES


ACM Computing Surveys, Vol. V, No. N, Article A, Publication date: January YYYY.
Augmentation Techniques for Mobile Cloud Computing: A Taxonomy, Survey, and Future Directions


Augmentation Techniques for Mobile Cloud Computing: A Taxonomy, Survey, and Future Directions


Augmentation Techniques for Mobile Cloud Computing: A Taxonomy, Survey, and Future Directions


