A Taxonomy of Computation Offloading in Mobile Cloud Computing

Nur Idawati Md Enzai and Maolin Tang *Senior Member, IEEE*

***Abstract*— The ability of cloud computing to provide almost unlimited storage, backup and recovery, and quick deployment contributes to its widespread attention and implementation. Cloud computing has also become an attractive choice for mobile users as well. Due to limited features of mobile devices such as power scarcity and inability to cater computation-intensive tasks, selected computation needs to be outsourced to the resourceful cloud servers. However, there are many challenges which need to be addressed in computation offloading for mobile cloud computing such as communication cost, connectivity maintenance and incurred latency. This paper presents taxonomy of the computation offloading approaches which aim to address the challenges. The taxonomy provides guidelines to identify research scopes in computation offloading for mobile cloud computing. We also outline directions and anticipated trends for future research.**

***Index Terms*— offloading; mobile; cloud computing**

I. Introduction

Cloud computing can be defined as a computing approach which provides dynamically scalable and often virtualized resources as services over the Internet [1]. Initially, cloud computing is utilized by fixed nodes or desktop users to make use of cloud resources. However, due to increasing portability and mobility of devices as well as dynamically changing network topology, simply adopting the wired cloud computing paradigm is not feasible. The distinct features of mobile devices and wireless communication networks need to be considered. On the other hand, the factor of limited resources (storage and processing power) of mobile devices boosts the requirement of cloud computing.

Consequently, the mobile devices need other resource providers to perform the execution of its mobile applications. Mobile cloud represents an infrastructure which could allow data storage and processing to occur outside the mobile device. The usage of mobile cloud allows execution of computer intensive applications on low resource mobile devices [2]. The availability of cloud computing services in a mobile environment to deal with resource limitations of mobile devices is defined as mobile cloud computing (MCC) in [3]. MCC is also explained as a concept that aims at using cloud computing techniques for storage and processing of data on mobile devices [4].

Differing to previous definitions which highlight the lack of resources in mobile devices, MCC is defined by [5] as optimization of an objective function which involves the execution of a mobile application within cloud platform. Examples of objective function are the application response time and energy consumption where the goal is to minimize the objective function.

Nevertheless, MCC is still based on fundamental cloud computing requirements which include on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service [6]. The plus side of mobile cloud computing compared to its base are its empowered mobile computing and a wide range of potential mobile cloud applications such as image processing, natural language processing, sharing GPS, sharing Internet access, sensor data applications, querying, crowd computing and multimedia search [2].

In summary, MCC provides mobile users with the data processing and storage services in the cloud [7]. Therefore, limitations of mobile devices especially the processing power and data storage can be leveraged. The battery life could also be prolonged by transferring the execution of computation intensive application to the cloud [8]. The process of moving computation to the cloud is called computation offloading.

Computation offloading is typically used to boost the computational capability of a resource constrained device for a single user. Computation offloading is also known as ”cyber foraging” and ”surrogate computing”. These terms are used interchangeably [9]. In our opinion, computation offloading is best described in [10]. Computation offloading is a mechanism where resource-intensive computations are migrated from a mobile device to the resource-rich cloud or server or nearby infrastructure. Computation offloading evolves from serving client-server paradigm to mobile systems and cloud computing. Client-server paradigm is still part of a cloud. However, cloud computing implies business, data stores, and other resources which are remotely hosted. Nonetheless, simply adopting the migration concept of client-server to cloud computing is not straightforward. Characteristics which are unique to cloud computing such as visualization and elasticity of cloud resources need to be taken into consideration. One of frameworks for computation offloading approaches consists of a partitioner, a profiler or a resource monitor and a solver or cost model [11]. A user agent on the mobile device and a server coordinator to handle the authentication and security is also included as one of offloading components in [5].

- The partitioner determines which portions of a computation to be offloaded. A computation can be modeled using a call graph that comprises of a set of nodes and a

N. I. Md Enzai is with School of Electrical Engineering and Computer Science, Queensland University of Technology, 2 George Street, Brisbane, Australia (email: nuridawati.mdenzai@student.qut.edu.au); M. Tang is with School of Electrical Engineering and Computer Science, Queensland University of Technology, 2 George Street, Brisbane, Australia (email: m.tang@qut.edu.au).

set of edges. The nodes could be software components, program modules, procedures, functions, objects, methods or threads based on the granularity level employed. The edges usually represent the interactions between nodes such as invocations. The edges could also be used to represent the maximum number of units of data that a channel can hold [12]. The identified partitions need to be determined if they could be offloaded. For instance, for a partition to be offloaded, it must not access I/O devices and hardware of a mobile device [13]. Common methods for partitioning are through static analysis and dynamic analysis [13], [14], [15], [16]. Static analysis is an offline method and dynamic analysis is conducted online. Though dynamic analysis is more accurate, it poses challenges such as complexity and extra overhead. Examples of analysis tools are Soot and JOrchestra [17], [18].

- The profiler measures and estimates the weight of the nodes and edges in the call graph. Weight is usually the cost of executions. The cost for nodes could be the execution time, energy consumption for execution, CPU cycles and memory used for execution [13], [14], [15], [16], [19]. Meanwhile, examples of weight for edges are: size of data to be transferred between two nodes, time taken to transfer data or state and energy consumed to transfer the state. Possible elements to be profiled are device, network and program [13], [14], [15], [16], [19]. The instrumentation and measurement process could also adapt to varying conditions for instance changing network bandwidth and intermittent connectivity [20] [21]. Output of profiling is then fed to the optimizer or solver. An example of power estimation model is PowerTutor [22].
- The solver aims to minimize cost of computation offloading such as energy consumption and execution time. Basically it leverages between computation and communication cost. Based on solution generated by the solver, nodes are allocated for execution either locally on the mobile device or remotely in the cloud. The distribution could be between a client and a server (single site), or a client to multiple sites or between multiple clients and multiple sites. For offloading to multiple destinations, it could be in parallel or sequential. On the cloud side, the sites could be in virtualized form or clones of client device [13], [14], [19].

Another framework includes component to assess whether it is favorable to perform computation or just execute it locally instead. There are four requirements to be satisfied for a computation to be favorable for offloading in terms of performance gain and energy efficiency. The requirements are: heavy computation, fast server, small data exchange and high bandwidth [9]. The working flow of computation offloading is summarized in Fig. 1.

We are aware that there is quite a number of literature review conducted on mobile cloud computing and computation offloading. Among recent works are: reviews

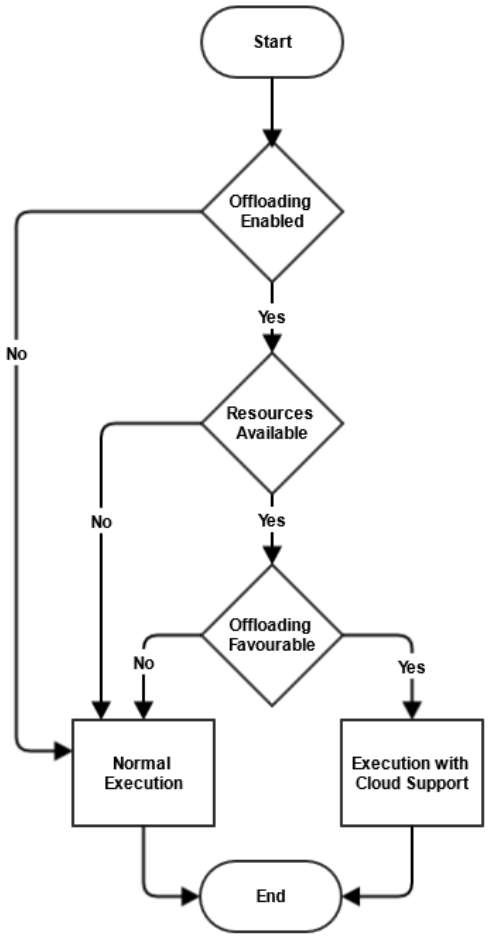

Fig. 1. Process of computation offloading [10]

of methods for boosting mobile cloud computing which includes offloading and surveys on computation offloading approaches for mobile systems up to year 2010 [5] [9]. Furthermore, reference [10] provides comprehensive overview of MCC in terms of architecture, offloading decision factors, classification and models. However, unlike our paper, the aspects of partitioning granularity and emerging application of distributed execution are not touched. This paper will focus on computation offloading since 2010 and specifically addresses the cloud computing environment instead of the whole mobile systems. Nevertheless, referring to older works is inevitable for knowledge background.

This paper classifies the computation offloading approaches for mobile cloud computing into six metrics namely: objectives, granularity, scheme, adaptation, distributed execution and communication. These categories are identified as common desired characteristics in a computation offloading approach.

II. OBJECTIVES

As mentioned in previous section, computation offloading framework comprises components such as partitioner, profiler

and solver. All of them will be conducted based on the objectives. For instance, if the objectives are to reduce energy and execution time, the related parameters of program, device and network will be estimated or profiled. The objectives are essential in determining the direction of design and development of a computation offloading framework. Partitioning decision is affected by computation offloading objectives.

Among popular objectives for computation offloading are to save energy and reduce execution time. This scope is addressed by [13], [14], [21], [23], [24], [25], [26]. Minimizing execution time and energy consumption are usually tackled together due to the relations as formulated below:

$$E = P \times T \tag{1}$$

where P represents power consumption, T is the time needed to execute computation and E is the formulated energy consumption. By reducing execution time, energy consumption can also be reduced.

The potential and applicability of computation offloading in saving energy has been explored in [27]. Kumar and Lu conclude that to make computation offloading worth it, the number of instructions (amount of computation) should be large, the size of data for offloading should be small and the network bandwidth should be large. In addition, the execution time and energy consumption for execution at mobile device should be larger than the time and energy taken to execute and transfer to the cloud [27].

Kumar and Lu also shows the formula for energy saving if the computation is executed at server instead of at mobile system as follows:

$$Pc \times \frac{C}{M} - Pi \times \frac{C}{S} - Pt \times \frac{D}{B} \tag{2}$$

Each parameter involved is detailed in Table I.

The trade-off between shortening execution time and extending battery life of mobile devices is further explored in [11]. Obviously offloading the application from mobile devices onto the remote cloud server can reduce execution time and save energy consumption. However, remote execution is not obligatory because processing on the cloud requires additional data communication, which may increase the execution time and the battery consumed by communication.

Based on these findings, Wu et al. argued that energy and time saving cannot be achieved simultaneously [11]. There must be some kind of trade-off between them. Their proposed computation offloading mechanism involves finding a server that satisfies constraints for server speedup which is represented by variable F. F is the factor for computation offloading based on whether to save energy or to save execution time.

Rather than directly dealing with minimizing execution time and energy usage, contributing factors such as maximizing throughput and data size reduction are also addressed. Yang et al. aim to maximize throughput as it determines the accuracy of many mobile data stream applications. Throughput is the number of units of input data the dataflow is able to process per second [12].

Meanwhile, the impact of data size on computation offloading performance has been studied in [28] and [29]. Only essential heap objects are transferred to reduce size. The characteristics of essential heap objects are: being referenced in the migrated thread, live and clean (not modified) [28]. CloneCloud also reduces the amount of data to send by using DEFLATE compression algorithm [13]. Reducing amount of data to be offloaded could affect the effectiveness of execution offloading and consequently improves mobile cloud computing performance as indicated in last part of Eq. 3 as follows.

$$E = P \times \frac{D}{B} \tag{3}$$

In the equation, $\frac{D}{B}$ formulates the time needed to transmit and receive the data. In effect, by reducing the data size, communication time and communication energy can also be reduced [27]. Similar to [28], an approach named Energy Efficient Task Scheduling (EETS) aims to reduce the amount of data transmission. The identification of suitable tasks for offloading is based on a tasks input/output data size and storage path. An energy usage model for each task is constructed for offloading decision to take place. EETS decides what type of task with particular size of data to be migrated to the cloud with respect to different input/output storage location [29].

Apart from achieving high throughput of processing the streaming data, scalability issues are also addressed [12]. To serve increasing number of users, computation instances on the cloud are able to be shared by multiple application or tenants. This sharing concept is also adopted in [30] by taking advantage of the scenario where the same code components which can be accessed by different users running the same or different applications could also be reused and shared data can be cached on the remote platform. Data mining techniques are suggested to detect potential data sharing across multiple applications as well as to construct suitable scheduling algorithms for this type of sharing. Even though sharing among multiple applications may enhance offloading performance, security concerns need to be anticipated if multiple users are involved. Users need to have control and awareness of what happens to their personal data which is stored in their mobile devices. Users permission is a must for sharing private data [2].

Kosta et al. also focus on scalability as in [12] by implementing parallel execution. This parallelization approach considers interval of input values for offloaded tasks distribution execution. In fact, energy consumption and execution time can be reduced as well [19]. Instead of making individual offloading decision for each method, [15] finds the offloading and integrating points for the whole program with all methods. This approach is based on the observation that when a method is offloaded, the subsequent calls will be offloaded with a high chance. It aims to achieve partition

TABLE I
ENERGY SAVING EQUATION PARAMETERS

Parameter	Detail
C	Number of instructions required by computation
S	Speed of cloud server (instructions per second)
M	Speed of mobile system (instructions per second)
D	Data (bytes)
B	Network bandwidth
Pc	Power consumed by mobile system for computing (watts)
Pi	Idle power consumption (watts)
Pt	Power consumed for sending and receiving data (watts)

accuracy and fast decision making by largely reducing the partitioning computation on cloud. Addressing the same issue but with regards to multiple sites, Sinha and Kulkarni specified the requirement that all allocation sites that refer to the same variable be offloaded as a single unit. This reduces the number of variations of the code to the number of offloading sites (since a particular variable will always be offloaded to a known site) [16].

Another crucial offloading factor particularly for real-time applications is the latency for link between client and cloud. As shown in equation (2), network latency is affected by network bandwidth. The issue of Wide Area Network (WAN) latency in accessing the cloud by deploying cloudlet is specifically addressed in[31], [32]. The term cloudlet is introduced by [31]. A cloudlet can be defined as trusted, resource-rich computer or cluster of computers thats well-connected to the Internet and available for use by nearby mobile devices. Rather than relying on a distant cloud, a mobile devices resource poverty can be catered via a nearby resource-rich cloudlet. In effect, the offloading communication time can be reduced. The cloudlet architecture used in [32] is shown in Fig. 2.

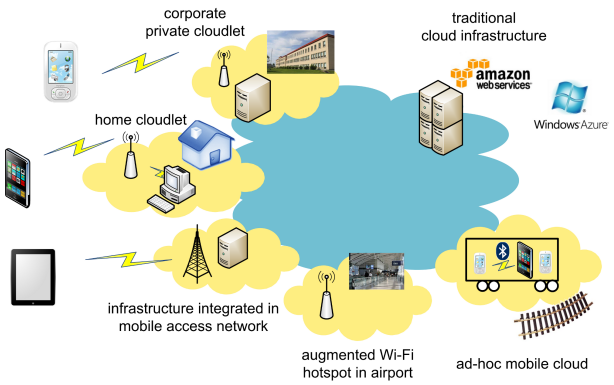

Fig. 2. Cloudlet architecture

Other than energy and execution time, Wu, Wang and Wolter also list price and storage as cost criteria for offloading decision. However, storage is not a major concern for offloading and price is not the scope of our study. Performance and robustness can also be enhanced through suitable offloading criteria with respect to types of application [11].

Whereas, a computation offloading framework named Cuckoo which specifically targets Android platform is proposed by [25]. Realizing the need for additional effort and skills by application developers to conduct computation offloading, Cuckoo provides very simple programming model that aims to tackle disconnection, support local and remote execution as well as packaging all code. Instead of focusing on Android platform, rich mobile applications are the target for Cloud framework in [33]. The main issues tackled in mobile cloud applications are complexity of application development and offline usability.

Meanwhile, instead of getting involved with computation offloading techniques, some researches indirectly address supporting mechanisms for offloading such as synchronization, multiple criteria decision analysis, software composition and real life scenarios implementation [34], [35], [36]. Even though the works assist in improving efficiency and performance of computation offloading in mobile cloud computing, the offloading techniques have not been directly addressed. Some works assumed that partitioning and outsourcing have been successfully conducted thus reducing the possibility of ensuring efficiency as a whole.

III. GRANULARITY

Computation offloading methods can be categorized into many ways, either in term of portion being offloaded or its granularity. In most cases, finer grained techniques involve partitioning meanwhile coarse grained techniques perform full migration. Fine-grained computation offloading techniques aim to reduce portion of data transmission and consequently able to save energy. However, partitioning process either conducted by programmer or remote execution manager may lead to additional overhead. Therefore, coarse-grained computation offloading approaches deal with this issue as well as reducing burden on programmer, yet still unable to resolve energy consumption concern [37].

A. Coarse-Grained

Virtual machine (VM) technology is utilized by mobile users to perform rapid instantiation of customized service software on a nearby cloudlet and then use that service over a wireless LAN. Specifically, dynamic VM synthesis

works by delivering a small VM overlay to the cloudlet infrastructure that already has base VM from which the overlay was derived. The infrastructure applies the overlay to the base to derive the launch VM, which starts executing in the precise state in which it was suspended [31].

Crisp interactive response, which is essential for seamless augmentation of human cognition, is easily achieved in this architecture because of the cloudlet's physical proximity and one-hop network latency. Using a cloudlet gives advantages in terms of energy and bandwidth usage. Though mobile devices manage to function as thin clients thus reducing computation offloading and battery usage on their side, substantial amount of time is needed to synthesize VM. This work has been the basis and reference for VM and cloudlet related researches later on. Nevertheless, it is a complete VM migration that requires a lot of state transfer which could cause more computation to be performed locally [38].

B. Fine-Grained

Instead of moving a complete virtual machine as done in [31] from the cloud to the cloudlet, a finer grained cloudlet concept that manages applications on a component level is outlined. These application components can be allocated among the cloudlets. The cloudlets can be formed with any LAN device with available resources dynamically. Due to its dynamic infrastructure, devices can join and leave the cloudlet at runtime [32]. Rather than running complete clone, an approach called Cuckoo executes a temporary clone restricted to only the service used by application. As a result, the cost of mobile device synchronization with an application clone in the cloud can be avoided [25].

However the issue of fine-grained offloading is already addressed before the concept of cloud computing and visualization comes into picture. Back in procedural programming paradigm era, the application granularity for offloading is procedure or function-level [23], [29]. When object-oriented paradigm emerged, class level is the initial choice for fine grained offloading [40] [41] and this granularity is still used recently [11] . Then, the granularity is refined to object level for more precision [42]. Wang and Franz develop an object relation graph (ORG) partitioning by employing the combination of static analysis and offline profiling techniques. Two-layer graph modelling is proposed to achieve unified strategy for different partitioning goals [42]. Object-level is also the choice of [16] [21] [24] [33] [41], [42]. Consequently with the development of cloud computing, method and threads level also became the choices for fine-grained offloading.

Cuervo et al. proposed and implemented a system called MAUI that offloads portions of a single application rather than as a whole. Profiling information of offloaded method is gathered after offloading to better predict whether future invocations should be outsourced or not. The decision optimization problem is determined by network connectivity to infrastructure, bandwidth and latency which are measured continuously to adapt to change. Based on the optimization, developer decides which application method to be offloaded. The remote server will invoke the method for offloading [14]. CloneCloud is another approach employing method-level offloading. Estimation of portion of the process to be executed is done through offline static analysis of different running conditions [14]. The granularity level used by MAUI and CloneCloud offloading framework have become main references for many works. Other works that use method-level approach are [13] [15] [19].

Although this method-level provides fine grained granularity in terms of control on what methods to execute remotely, it would require significant state transfer as each method gets called. Due to this reason, Shivarudrappa, Chen and Bharadwaj favour thread-level offloading approach. Thread level differs from method level in term of its execution migration points. Instead of restricting the migration points to entry and exit point of a method, thread level allows any point in a method provided it satisfies the constraints and optimization objectives [38]. Thread-level granularity is also selected for its offloading process in [12] [28] .

Granularity terms are not restricted to method, object, class, and function. Task is employed as partitioning granularity by [29], [39], [43], and the term component is used in [32], [33]. Meanwhile, service, segment and module are denoted as partitioning granularity in [25], [44], [45], [46], [47]. Nevertheless, these terms usually refer to different functionalities in an application.

From software perspective, to address the lack of software composition consideration in [13] and [14], μCloud provides a composition approach which allows configurable, modularity and flexibility features in applications as well as reusability of independent software components.

IV. SCHEME

As mentioned in the introduction section, most computation offloading approaches partition program before the stages of profiling and optimization. There are two ways of implementing partitioning either static or dynamic. Static partitioning is done during development meanwhile dynamic partitioning is conducted during execution [9]. Static partitioning is implemented by [16] [24]. On the other hand, dynamic or automatic partitioning is employed by [12], [13], [14], [19], [31], [32]. Dynamic partitioning incurs overhead as it is continuously done to obtain the latest information. The beneficial features and drawbacks of both methods are described in [7] and summarized in Table II.

The pros and cons between static and dynamic partitioning are also highlighted in [37]. Static partitioning could not adapt to varying network conditions efficiently and also places more responsibility on programmers. A partitioning mechanism which automatically computes estimated partitioning solution is more suitable. However, extra computing cost for dynamic partitioning is still an issue. Nonetheless, dynamic partitioning is a common method to incorporate adaptive feature in computation offloading framework.

Intervention of programmer as implemented in [14] also indicates that computation offloading is done manually in contrast to automatic offloading which does not require pro-

TABLE II
STATIC PARTITIONING VS. DYNAMIC PARTITIONING

Partitioning Decision	Advantage	Disadvantage
Static	Low overhead during execution	Beneficial only when the parameters can be accurately predicted in advance
Dynamic	Adapt to different runtime conditions	High overhead during execution

grammer annotation to identify methods to be offloaded [13], [19].

V. Adaptation

Adaptation means to take into account or consider different program execution contexts or instances. It will lead to quite different optimal program partitioning decisions as proven in [39]. Realizing this need, computation offloading approaches tackle various adaptation scopes such as varying bandwidth, network connectivity, workloads, deadlines of tasks and heterogeneous architectures.

ThinkAir adapts to varying bandwidth and connectivity changes during runtime. On-demand resource allocation based on different computational power based on workload and deadlines for tasks is applied through VM resource scaling. Data are collected to decide which method to be offloaded, but if it is recognized for the first time, the quality of connection becomes the decision factor. It also utilizes a virtualization environment to allow the system to be deployed where needed whether on a private or commercial cloud [19]. Adaptation to varying bandwidth is also addressed in [21], [24].

Even though Niu, Song and Atiquzzaman did not directly address offloading techniques, three partitioning models which considers bandwidth as a variable in the mobile environment are proposed. Each model has different objective with regards to execution time optimization, energy optimization and combination of both [24]. Reference [21] handles varying bandwidth by keeping track of current bandwidth at each encountered nodes during the cost minimization process. Critical bandwidth is set to be the threshold for finding optimal partitioning for offloading.

MAUI runs application profiling continuously to get up to date cost estimation of each method. It is able to execute the same code in different CPU architectures. Since mobile users may move in and out of MAUI's server range, optimization problem is resolved periodically to adapt to network changes [14].

CloneCloud adapts application partitioning to different environments [13]. Yang et al. handle adaptation at both mobile device and the cloud. First, on the mobile side, it needs to handle the wide variations and dynamic changes in network conditions and local resource availability. In order to achieve high performance, the decision of which units of computation should be moved to the cloud has to be made adaptive to the changes in mobile environments. Second, the cloud side needs to handle the unpredictable and varying load from multiple mobile clients of the application [12].

Another way to adapt to changing environment is through prediction as done in [20], [48]. By extending CloneCloud offloading framework, Shi et al. address the challenge of mapping computations onto nodes with an assurance that the necessary code and data can be delivered and the results received in time due to varying connectivity. The allocation algorithms are designed to cater three types of intermittent connectivity: predictable connectivity with control channel, predictable connectivity without control channel, and unpredictable connectivity [20]. These algorithms can also be customized for energy optimization objective. This work is further extended in [48] by catering the needs of different application requirement by setting different thresholds based on their properties.

VI. Distributed Execution

Computation offloading which involves a mobile device to a single server in the cloud is not a realistic cloud computing scenario [21]. It is more common for an application to be distributed among several sites. Besides, data is usually located in the clouds in distributed manner. For instance, an application needs to compare set of images from one server with set of images with another server. To perform this task, the mobile application might first retrieve all images from a server and extract appropriate features from the images. Then, by using pattern matching technique the photos are compared with set of photos which resides in another server [16].

Offloading to multiple servers could also increase server speedup by parallelizing the application computation [27]. This is illustrated in the equation below which is modified from Eq. 1 with assumption that the server is F times faster:

$$Pc \times \frac{C}{M} - Pi \times \frac{C}{F \times M} - Pt \times \frac{D}{B} \tag{4}$$

Ou, Yang and Liotta paved the way for multiple sites offloading in [41] and its framework is referred by [16]. Sinha and Kulkarni also argued that there is emerging need for application accesses to be distributed among several servers in contrast to previous works which concentrated on single mobile device offloading computation to a single server. Some challenges need to be addressed in term of partitioning algorithm development. The algorithm should be able to divide a program between multiple possible execution sites and takes into account distinct functionalities from site to site. Furthermore, the computation offloading should be conducted at the object level for allowing objects of the same class to be offloaded to different servers [16].

To reduce energy consumption, Energy-Efficient Multisite Offloading Algorithm (EMSO) also offloads part of computation to multiple remote servers or destinations [21]. Moreover, differentiated sites is also considered as in [16]. However, EMSO does not perform comparison study with virtualization methods. The general visualization of multiple sites offloading is depicted in Fig. 3.

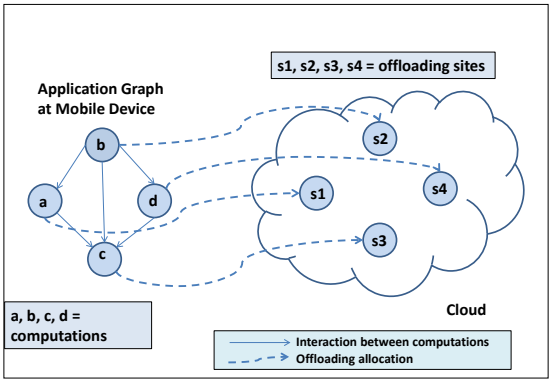

Fig. 3. Computation offloading with multiple sites

Another aspect for distributed execution involves multiple users as well as handled by [12], [19], [30], [43]. ThinkAir automatically splits and distributes tasks to multiple virtual machines. A framework is designed to facilitate the partitioning and execution of mobile data stream applications which requires parallel execution of different approaches onto the streaming data. To address growing number of mobile users, it also supports efficient utilization of cloud resources [19].

On the other hand, by utilizing mobile agent, a dynamic performance optimization framework for mobile cloud computing is proposed in [50]. The framework consists of an execution manager which is responsible for making the decision on where to execute the application partitions. A list of most promising cloud hosts (VM instances) is obtained from cloud directory service. A cost model which is based on execution time is utilized by the execution manager to make offloading decisions for each offloadable partition.

Cloud resources constraint is also another concern related to multiple sites offloading. The literatures which consider the cloud resources are constrained characterize the constraints in various aspects such as number of available servers, number of resources and number of cloud racks [49], [51], [52].

Parallelization is also an issue which is related to multiple destinations offloading. Kosta et al. also provide method-level computation offloading and enhances the capability of mobile cloud computing by parallelizing method execution in the cloud through multiple VM images. Scalability is expected to be enhanced by parallelizing the offloaded services as well as providing automatic partitioning on mobile applications [19]. This feature is also desired and mentioned in [32] where Verbelen et al. consider multiple places for remote execution either within the cloudlet or in other cloudlets for future work.

Another issue pertaining to parallel execution is the task distribution. Soyata et al. address task distribution through parallelism upon multiple cloud servers given heterogeneous communication latencies and compute powers of cloud servers at various locations [53]. To achieve energy efficiency, Yao et al. present a task allocation algorithm which is based on a tasks input/output datas size and storage path [29]. However, Soyata et al. argue that the task distribution algorithms for different cloud servers should be more generic thus able to cater the constraints of the mobile devices, cloudlets and the servers. The order of modules invocations involving multiple destinations is also a concern [5].

VII. Communication

Computation offloading to the cloud definitely requires communication between client and cloud. As a result, communication cost incurred must be included in offloading decision. Sinha and Kulkarni associate communication cost with movement of data and necessary messages in case parts of the application reside on different hosts varies depending on the hosts. Since offloading to multiple servers are applied, there are two types of communication involved; namely client to server and server to server. Communication between two cloud-resident servers is assumed to be faster than communication between the mobile device and the cloud [16].

Among parameters considered in evaluating communication cost are transmission delay, energy and network bandwidth. Zhang et al. consider delay in transferring time estimation [15]. Delay and energy are the concern for transmission cost in [24] and network latency is given attention in [31], [32].

High amount of offloaded data also affects the communication cost. This motivates [28] and [29] to reduce data size for offloading and leads to [16] devising strategy of moving computation to the data instead of transmitting the data over the network.

The network characteristics can be estimated through network profiling [44]. This approach is utilized by MAUI and CloneCloud [13], [14]. Profiler is also used in [12] to continuously monitor wireless network bandwidth. Latency and bandwidth characteristics are estimated in MAUI and CloneCloud to decide for future offloading. Optimization problem is also resolved periodically based on the network changes. MAUI approach is incorporated in [19] and CloneCloud method is adopted by [20] [28]. Meanwhile, EMSO model considers bandwidth changes of wireless network to estimate the communication cost [21].

VIII. Open Issues

Having reviewed various existing computation offloading methods and strategies, this section identifies some emerging computation offloading issues in mobile cloud computing. Heterogeneity of offloading sites: Computation offloading

TABLE III
SUMMARY OF EXISTING COMPUTATION OFFLOADING METHODS

Method	Objective	Granularity	Scheme	Distributed	Adaptive
[39]	N/A	Task	Automatic	No	Yes
[41]	To relieve memory, CPU usage and bandwidth constraints	Class	Automatic	Yes	Yes
[40]	Minimize overall response time	Class	Automatic	Yes	Yes
[42]	Minimize energy consumption and execution time	Object	Automatic	No	No
[31]	Minimize network latency	N/A	Automatic	N/A	No
[48]	Address heterogeneous environment	Module	Automatic	No	Yes
[13]	Minimize energy	Thread	Automatic	No	Yes
[15]	High partition accuracy	Method	N/A	N/A	N/A
[16]	Minimize computation time Minimize local storage needs Minimize battery usage	Object	Manual	Yes	Yes
[33]	Energy optimization	Component	Automatic	Yes	Yes
[11]	Reduce time and save energy	N/A	Automatic	No	Yes
[12]	Maximum speed or throughput	Method	Automatic	Yes	Yes
[19]	Reduce execution time and energy consumption through parallel execution	Method	Automatic	Yes	Yes
[20]	Speedup computing and conserve energy under intermittent connectivity scenarios	Thread	Automatic	No	Yes
[25]	Reduce energy consumption and increase the speed of computation-intensive operations	Service	Manual	Yes	Yes
[30]	Utilize relation among multiple applications	N/A	N/A	Yes	No
[32]	minimize network latency	Component	Automatic	No	Yes
[47]	Minimize overall response time	N/A	Automatic	Yes	Yes
[49]	Improve performance and reduce energy consumption	Service	Automatic	No	Yes
[50]	Better performance and longer battery life	Service	Automatic	No	No
[51]	Decrease response time and energy consumptions	Segment	Automatic	No	No
[21]	Minimize energy consumption as the network bandwidth changes	Object	Automatic	Yes	Yes
[24]	Reduce execution time and energy consumption	Object	Automatic	Yes	Yes
[28]	Reduce transferred data size	Method	Automatic	No	Yes
[29]	Reduce transferred data size	Task	Automatic	Yes	Yes
[44]	Minimize energy consumption and execution time	N/A	Automatic	Yes	Yes
[46]	Achieve high performance while preserving fairness	Task	Automatic	Yes	Yes
[43]	Reduce energy consumption and minimize average application delay	N/A	Automatic	Yes	Yes

with multiple users and destinations may incur load balancing and scheduling problem [12]. A more generic task distribution algorithm that considers resources and requirements of the mobile devices, cloudlets and servers is needed [5]. Different features of offloading sites can also be more refined [16].

Optimal partitioning: To improve the efficiency of remote execution, only the required data needs to be offloaded and redundant data transfer should be minimized. By using static analysis tool, to further reduce the amount of state transferred, the variables that are actually referenced in the remote method must be determined [14]. CloneCloud suggests to further refine differentiations depending on calling stack, methods and arguments [9]. To allow more precise computation offloading, Sinha and Kulkarni argue that this can be achieved by considering objects that are fully enclosed by other objects to be part of the enclosing object rather than requiring a separate offloading decision for them [16]. Meanwhile ThinkAir proposes combining static code analysis with data caching. The former eliminates the need to send and receive data that is not accessed by the cloud. The latter ensures that unchanged values need not be sent, in either direction, repeatedly. This could be further combined with speculative execution to explore alternative execution paths for improved caching [19].

Latency: On-demand resource allocation allows dynamic control of resources but it introduces latency by resuming, starting, and synchronizing among the virtual machines

(VMs) especially when the number of VMs to be resumed concurrently is high. A user may also have different QoS requirements (e.g. completion time) for different tasks at different times, therefore the VM manager needs to dynamically allocate the number of VMs to achieve the user anticipations [19].

Connectivity: More advanced profiler that is able to measure the network quality is needed, instead of just testing if there's a connection, and selecting the best server to offload its task when multiple servers exists. It should also be capable of adjusting its offload policy according to the overall evaluation on the device, program and network profiles [38]. To our knowledge, current implemented mechanisms to overcome effect of connectivity loss to the cloud simply opt to resume execution locally [13], [19], [25]. However, in the event of intermittent connectivity, local execution may not be the optimal measure for best performance.

Complexity: To enhance performance of computation offloading, processes involved such as profiling, partitioning and solving may become more complex to implement and could incur additional overhead. ThinkAir experiences extra synchronization overhead between primary server and secondary server even though the architecture is much better compared to connecting mobile device to every single server [19]. Meanwhile, MAUI profiling process consume processing power, memory and energy of the smartphones [14]. Therefore, mechanism of leveraging the degree of complexity and desired performance is crucial.

Among the research questions that can be raised are:

- How to characterize the relationship of granularity level with respect to type of requested workload by user?
- How to overcome the challenges and limitations brought by parallelization in multi-site offloading?

Taking into account the emerging issues in computation offloading for mobile cloud computing, the reviewed approaches are summarized in Table III above.

IX. Conclusion

This paper has reviewed and categorized recent researches related to computation offloading for mobile cloud computing. The classification is based on the desired features for a computation offloading approach. We have also described the solutions and measures implemented by existing computation offloading methods to achieve and enhance each feature. The approaches are presented in taxonomy form. This taxonomy is beneficial in highlighting prospective areas for further research. In addition, we have identified and discussed emerging computation offloading issues and challenges in mobile cloud computing.

References

[1] M. R. Prasad, J. Gyani, and P. Murti, Mobile Cloud Computing: Implications and Challenges, *Journal of Information Engineering and Applications*, vol. 2, pp. 7-15, 2012.

[2] N. Fernando, S. W. Loke, and W. Rahayu, Mobile cloud computing: A survey, *Future Generation Computer Systems*, 2012.

[3] M. Schring, Mobile cloud computing-open issues and solutions, *15th Twente Student Conference on IT*,2011.

[4] S. Chetan, G. Kumar, K. Dinesh, K. Mathew, and M. Abhimanyu, Cloud computing for mobile world, *National Institute of Technology*, Calicut, 2010.

[5] T. Soyata, H. Ba, W. Heinzelman, M. Kwon, and J. Shi, Accelerating Mobile-Cloud Computing: A Survey, in *Communication Infrastructures for Cloud Computing*, H. T. M. a. B. Kantarci, Ed., ed: IGI Global, 2013, pp. 175-197.

[6] N.-M. Peter and G. Timothy, The NIST definition of cloud computing (draft), *NIST Special Publication*, pp. 800-145, 2011.

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

[8] A. Klein, C. Mannweiler, J. Schneider, and H. D. Schotten, Access schemes for mobile cloud computing, in *Proceeding of the Eleventh International Conference on Mobile Data Management*, 2010, pp. 387-392.

[9] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, A Survey of Computation Offloading for Mobile Systems, *Mobile Networks and Applications*, vol. 18, pp. 129-140, 2013.

[10] A. Khan, M. Othman, S. Madani, and S. Khan, A Survey of Mobile Cloud Computing Application Models, *Communications Surveys & Tutorials, IEEE*, vol. PP, pp. 1-21, 2013.

[11] H. Wu, Q. Wang, and K. Wolter, Tradeoff between Performance Improvement and Energy Saving in Mobile Cloud Offloading Systems," in *Communications Workshops (ICC), 2013 IEEE International Conference on*, 2013, pp. 728-732.

[12] L. Yang, J. Cao, S. Tang, T. Li, and A. T. Chan, A Framework for Partitioning and Execution of Data Stream Applications in Mobile Cloud Computing, in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, 2012, pp. 794-802.

[13] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, Clonecloud: elastic execution between mobile device and cloud, in *Proceedings of the sixth conference on Computer systems*, 2011, pp. 301-314.

[14] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, et al., MAUI: making smartphones last longer with code offload, in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, 2010, pp. 49-62.

[15] Y. Zhang, H. Liu, L. Jiao, and X. Fu, To offload or not to offload: An efficient code partition algorithm for mobile cloud computing, in *Cloud Networking (CLOUDNET), 2012 IEEE 1st International Conference on*, 2012, pp. 80-86.

[16] K. Sinha and M. Kulkarni, Techniques for fine-grained, multi-site computation offloading, in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, 2011, pp. 184-194.

[17] P. Lam, E. Bodden, O. Lhotk, and L. Hendren, The Soot framework for Java program analysis: a retrospective, in *Cetus Users and Compiler Infastructure Workshop (CETUS 2011)*, 2011.

[18] E. Tilevich and Y. Smaragdakis, J-Orchestra: Enhancing Java programs with distribution capabilities, *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 19, p. 1, 2009.

[19] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading, in *INFOCOM, 2012 Proceedings IEEE*, 2012, pp. 945-953.

[20] C. Shi, M. H. Ammar, E. W. Zegura, and M. Naik, Computing in cirrus clouds: the challenge of intermittent connectivity, in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 23-28.

[21] Y. Liu, An Energy-Efficient Multisite Offloading Algorithm for Mobile Devices, *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.

[22] Z. Yang, PowerTutor-A Power Monitor for Android-Based Mobile Platforms, EECS, *University of Michigan*, retrieved September, vol. 2, 2012.

[23] Z. Li, C. Wang, and R. Xu, Computation offloading to save energy on handheld devices: a partition scheme, in *Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems*, 2001, pp. 238-246.

[24] J. Niu, W. Song, and M. Atiquzzaman, Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications, *Journal of Network and Computer Applications*, 2013.

[25] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, Cuckoo: a computation offloading framework for smartphones, *Mobile Computing, Applications, and Services*, pp. 59-79, 2012.

[26] Y. Wen, W. Zhang, and H. Luo, Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones, in *INFOCOM, 2012 Proceedings IEEE*, 2012, pp. 2716-2720.

[27] K. Kumar and Y.-H. Lu, Cloud computing for mobile users: Can offloading computation save energy?, *Computer*, vol. 43, pp. 51-56, 2010.

[28] S. Yang, Y. Kwon, Y. Cho, H. Yi, D. Kwon, J. Youn, et al., Fast Dynamic Execution Offloading for Efficient Mobile Cloud Computing, in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2013, p. 22.

[29] D. Yao, C. Yu, H. Jin, and J. Zhou, Energy Efficient Task Scheduling in Mobile Cloud Computing, in *Network and Parallel Computing*, ed: Springer, 2013, pp. 344-355.

[30] C. Mei, D. Taylor, C. Wang, A. Chandra, and J. Weissman, Sharing-aware Cloud-based Mobile Outsourcing, in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, 2012, pp. 408-415.

[31] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, The case for vm-based cloudlets in mobile computing, *Pervasive Computing, IEEE*, vol. 8, pp. 14-23, 2009.

[32] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, Cloudlets: Bringing the cloud to the mobile user, in *Proceedings of the third ACM workshop on Mobile cloud computing and services*, 2012, pp. 29-36.

[33] V. March, Y. Gu, E. Leonardi, G. Goh, M. Kirchberg, and B. S. Lee, ?Cloud: Towards a New Paradigm of Rich Mobile Applications, *Procedia Computer Science*, vol. 5, pp. 618-624, 2011.

[34] E. Lagerspetz and S. Tarkoma, Mobile search and the cloud: The benefits of offloading, in *Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2011 IEEE International Conference on, 2011, pp. 117-122.

[35] H. Wu, Q. Wang, and K. Wolter, Methods of cloud-path selection for offloading in mobile cloud computing systems, in *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, 2012, pp. 443-448.

[36] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing, in *Proc. of IEEE INFOCOM*, 2013.

[37] X. Ma, Y. Cui, L. Wang, and I. Stojmenovic, Energy optimizations for mobile terminals via computation offloading, in *Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on*, 2012, pp. 236-241.

[38] D. Shivarudrappa, M. Chen, and S. Bharadwaj, "COFA: Automatic and Dynamic Code Offload for Android."

[39] C. Wang and Z. Li, Parametric analysis for adaptive computation offloading, *ACM SIGPLAN Notices*, vol. 39, 2004, pp. 119-130.

[40] K. Yang, S. Ou, and H.-H. Chen, On effective offloading services for resource-constrained mobile devices running heavier mobile Internet applications, *Communications Magazine, IEEE*, vol. 46, 2008, pp. 56-63.

[41] S. Ou, K. Yang, and A. Liotta, An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems,in *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, 2006, pp. 10 pp.-125.

[42] L. Wang and M. Franz, Automatic Partitioning of Object-Oriented Programs for Resource-Constrained Mobile Devices with Multiple Distribution Objectives, in *Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE International Conference on*, 2008, pp. 369-376.

[43] J.-L. Chen and F.-J. Wang, Flow Analysis of Class Relationships for Object-Oriented Programs, *Journal of Inf. Sci. Eng.*, vol. 16, 2000, pp. 619-647.

[44] B.-G. Chun and P. Maniatis, Dynamically partitioning applications between weak devices and clouds, in *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, 2010, p.7.

[45] D. Kovachev, Y. Tian, and R. Klamma, Adaptive Computation Offloading from Mobile Devices into the Cloud, in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, 2012, pp. 784-791.

[46] H.-Y. Chen, Y.-H. Lin, and C.-M. Cheng, COCA: Computation Offload to Clouds Using AOP, in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, 2012, pp. 466-473.

[47] S. Yang, Manageable Granularity in Mobile Application Code Offloading for Energy Savings, in *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, 2012, pp. 611-614.

[48] C. Shi, P. Pandurangan, K. Ni, J. Yang, M. Ammar, M. Naik, et al., IC-Cloud: Computation Offloading to an Intermittently-Connected Cloud, *available at smartech.gatech.edu*,2013.

[49] L. Yang, J. Cao, and H. Cheng, Resource Constrained Multi-user Computation Partitioning for Interactive Mobile Cloud Applications, *Technical report*, Dept. of Computing, Hong Kong Polytechnic Univ., 2013.

[50] P. Angin and B. Bhargava, An Agent-based Optimization Framework for Mobile-Cloud Computing, *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 4, 2013, pp. 1-17.

[51] X. Wei, J. Fan, Z. Lu, and K. Ding, Application Scheduling in Mobile Cloud Computing with Load Balancing, *available at downloads.hindawi.com*, 2013.

[52] P. Mousicou, C. X. Mavromoustakis, A. Bourdena, G. Mastorakis, and E. Pallis, Performance evaluation of dynamic cloud resource migration based on temporal and capacity-aware policy for efficient resource sharing, in *Proceedings of the 2nd ACM workshop on High performance mobile opportunistic systems*, 2013, pp. 59-66.

[53] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture, in *Computers and Communications (ISCC), 2012 IEEE Symposium on*, 2012, pp. 59-66.