A comprehensive survey of code offloading mechanisms for mobile cloud computing

Mohammad Erfan* , Bidoura Ahmad Hridita, Mohammad Shoyaib, and Md. Shariful Islam

***Abstract*—Resources, battery lifetime and storage capabilities of mobile devices are affected by the compute intensive, resource intensive or energy drain applications. These limitations of the mobile devices may be mitigated with the help of cloud computing by delegating the energy drain or computing intensive tasks to more resourceful servers and receiving the result from the server. This process (a.k.a code offloading) helps to increase performance and reduce energy consumption. We present a detailed survey of the state of the art code offloading mechanisms for mobile based applications to better understand the domain of code offloading. The paper provides a high level design architecture, procedure for code offloading and presents a comparison of existing mechanisms based on the level of execution, level of profiling, security requirements, code profiler and offloading adaptation context. This paper also highlights major research challenges related to code offloading and provide recommendations to take better decision on code offloading in order to build more powerful mobile applications.**

I. INTRODUCTION

The mobile applications development and deployment area are expanded by most exciting technology Mobile Cloud Computing (MCC) with techniques such as code offloading. However, mobile devices have limited battery life, limited storage, inconsistent network bandwidth which make it hard to execute compute intensive and energy drain applications on the mobile devices. Mobile devices are gaining popularity in todays life because of its small size, high computing capabilities, communication capabilities and less carrying overhead insignificant of place and time. The limitations of the mobile devices can be mitigated by migrating either application or part of it to remote computing device to gain benefits using MCC. MCC executes high computational or energy hungry applications to the cloud which allows the mobile users to use the cloud as infrastructure as a service (IAAS), software as a service (SAAS) and platform as a service (PAAS) at low cost and on demand fashion. Cloud computing makes it possible to augment mobile devices capabilities in terms of energy, storage, computation, data safety and security [1].

Several types of mobile applications in recent days such as image processing, voice recognition, object recognition, translation and m-gaming etc. have limiting factor to execute in mobile devices which introduce high computation problems, energy consumption and huge execution cost [2]. These type of applications can be executed on mobile devices by migrating huge computational tasks to nearby servers using MCC which is usually known as *computation offloading or code offloading*. Mobile applications compute intensive or energy drain methods, threads or classes are assigned to resourceful server referred to as code offloading for performance improvement [3], [4], increasing the battery life [5] and reducing the cost of execution. The effectiveness of an offloading system is determined by its ability to correctly identify which tasks are candidate for offloading and where to execute the offloadable task either locally or remotely.

Code offloading or computation offloading mechanisms consist of identifying which parts of the codes are offloaded referred as what to offload. Different types of environmental context such as network bandwidth, server load, execution time and execution cost for optimally taking offloading decision are referred as when to offload. Static or dynamic decision of code partitioning based on the resource and energy consumption means how to offload. Identification of the remote cloud server for the offloaded code means where to offload. Code offloading is beneficial when the computing device saves energy or improves per-

* Corresponding author.

Mohammad Erfan, Bidoura Ahmad Hridita, Mohammad Shoyaib, and Md. Shariful Islam are with the Institute of Information Technology, University of Dhaka, Bangladesh. e-mail: {bit0326, bit0316, shoyaib, shariful}@iit.du.ac.bd.

Manuscript received January 25, 2016; revised March 03, 2016.

formances; and counter-productive when the device wastes energy for executing a task remotely rather than locally.

The purpose of this paper is to familiarize the researcher in the computation offloading research area for mobile based applications. A comprehensive survey is performed on existing common code offloading approaches used to make offloading decisions that classifies the code offloading approaches: thread level, method level, and data size level considering different variable parameters such as network connectivity, execution time, server load, and execution cost. This paper serves a generic code offloading mechanism, high level design architecture and a generic flow chart for code offloading. The paper compares the state of the art code offloading approaches on the basis of level of execution, level of profiling, security requirements, code profiling, and offloading adaptation context. This paper also includes major findings and discussions to take optimal offloading decisions. The paper concludes with the open research challenges for researchers on the field of code offloading.

The paper is organized as follows: Section 2 describes mobile cloud computing and code offloading mechanism. Section 3 describes the existing code offloading mechanisms. Section 4 describes the comparison of existing code offloading mechanisms to take better decisions. Major findings on the code offloading are described in Section 5. Section 6 describes the open research challenges and Section 7 concludes the paper.

II. MOBILE CLOUD COMPUTING AND CODE OFFLOADING

This section gives a brief description, architecture and mechanism related to MCC and code offloading. In order to provide basic understanding to the researchers, the paper provides a generic flow chart and a high level design for code offloading.

A. Mobile Cloud Computing

Mobile cloud computing uses the architecture of cloud computing for computing intensive and energy hungry applications to make it possible to execute on the mobile. MCC helps mobile users to get ubiquitous access to available resources provided by service providers with the purpose of maximizing battery life, data storage, inconsistent network bandwidth, processing power and data safety. MCC removes the limitations of mobile devices by incorporating mobile devices with mobile computing, cloud computing and the network technologies. It also makes it possible to develop and execute the mobile applications at lower cost and makes it easy for developers to acquire new technology on demand basis.

Nowadays, mobile users want to execute PC like applications which are resource hungry on their mobile devices. The development of hardware technologies make the mobile devices smaller in size and make it capable of executing applications in mobile. Image processing, voice recognition and m-gaming applications are executable in todays mobile device but their energy consumption decreases battery lifetime drastically. The cloud providers serve their services as SAAS, IAAS and PAAS to the mobile users and tasks result are back on to the device as displayed in Fig. 1.

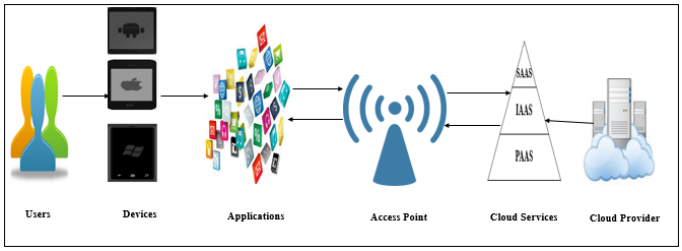

Fig. 1: Mobile Cloud Computing Architecture

B. Code Offloading

The process of delegating methods, classes, threads or data which are resource intensive to the remote system in order to improve the performance of mobile device, increasing the battery life called *Code Offloading*. The code offloading technique includes to identify the offloadable part of the application for remote execution. It takes decision by considering the environmental context such as network bandwidth, server load, execution time, and execution cost. The system then partition the code statically [6], [7] or dynamically [8]–[12] and optimally detect the offloadable task for remote execution. Finally the mechanism detect the remote server for seamlessly executing the offloaded task. The remote executable methods selected by optimazation solver are delegated to remote server. The remote server may communicate to another server for seamlessly executing the request. After completing the application execution, the remote server sends the computing results back to the mobile

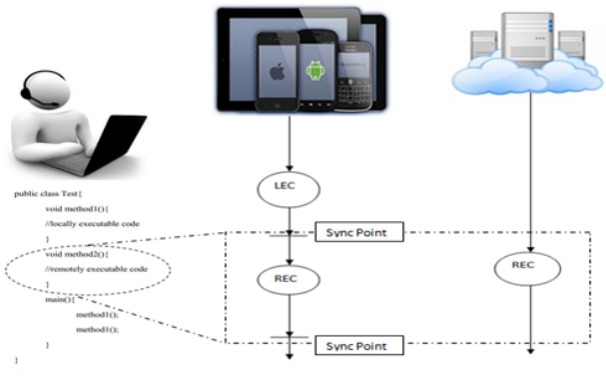

Fig. 2: Code Offloading Mechanism

device.

Mobile applications consist of methods which require little or more computation depending on the resource usage of mobile devices. Methods that uses mobile sensors, internal IO devices or sensible executable data where re-execution hamper the actual results are always executed locally. The rest of the methods executed either locally or remotely based on the variable parameters and remote execution saves energy or decreases execution time.

Mobile application methods with different execution time and energy consumption where remote executable methods are selected by the optimization solver. The selectable methods are delivered to the remote server and the rest of the methods are executed on the mobile. The remote executable tasks are executed on the resourceful cloud and the results are sent back to the devices. Finally the results are synchronized to the mobile device which is depicted in Fig. 2.

At the start of the offloading process, local and remote executable methods are initialized as local or remote executable. Different types of variable parameters such as network, device, server and program information are used as input to the offloading solver to take the offloading decision. If offloading does not save energy or improve performance, it executes the methods locally. Otherwise, the device sends it to the remote resources to completely execute the execution. The general flow chart for code offloading framework is depicted in Fig. 3.

The state of the art code offloading framework have same type of high level design architecture which is displayed in Fig. 4. The code offloading design architecture have communication controller,

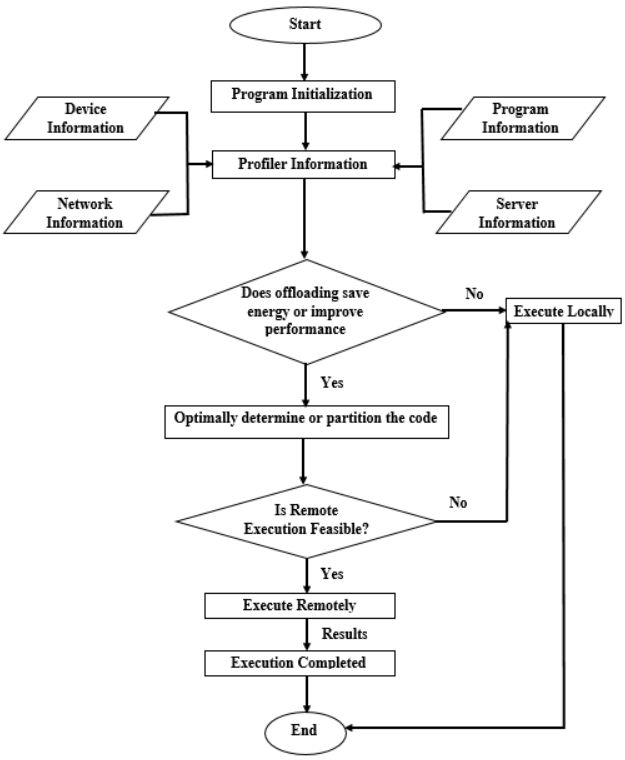

Fig. 3: Code Offloading Flowchart

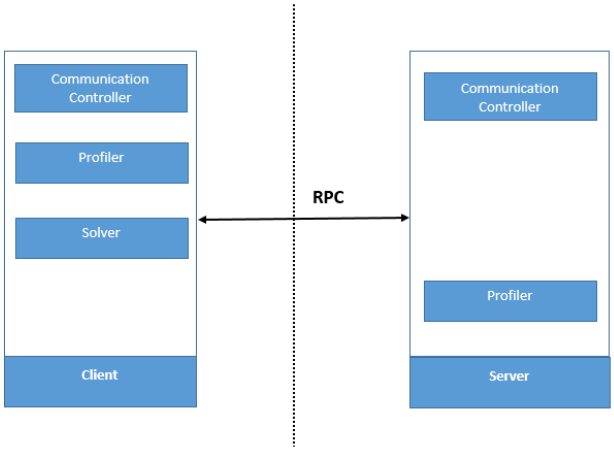

Fig. 4: High Level Design for Code Offloading

profiler and optimization solver on mobile device where the profiler information serves as the input to the solver. The communication controller controls the connection between the device and server. The main purpose of the controller is to deliver the tasks to the server, receive the results and update the changes. The communication manager re-executes the code locally for any type of failures in the completeness of the

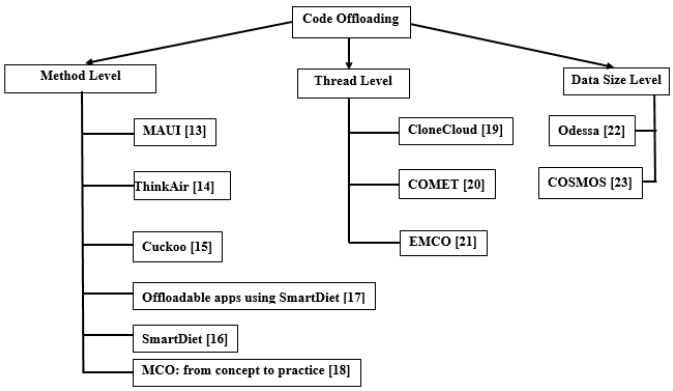

Fig. 5: Code Offloading Categories

application. The profiler stores device, network, and program and server information to take offloading decision. The device profiler collects device status, battery life and CPU usage. The network profiler collects network bandwidth, latency information. The program profiler stores execution time, size of execution and energy consumption information, and the server profiler stores server workload information. The optimization solver is used to optimally select the remote executable methods for an application to maximize the performance and energy saving.

III. CODE OFFLOADING MECHANISMS

This section describes a comprehensive survey of the existing code offloading mechanisms. There are many research works related to the cloud computing, but survey on specifically code offloading is limited. The code offloading mechanisms are designed to run the compute intensive and energy drain applications to remote servers aiming to inflate the computation capabilities and energy efficiency.

In code offloading, tasks can be method, class, thread or data depending on the partitioning level. The state of the art code offloading mechanisms can be classified into three categories banamed method level granularity, thread level granularity and data size level granularity which are depicted in Fig. 5. In the following section, the method level, thread level and data size level code offloading mechanisms are described followed by a comparative study.

A. Method level granularity

In code offloading, method level granularity refers to deliver the computational or resource intensive methods to the remote server to save CPU cycles. The code offloading mechanism partitioned the code at fine grain. It is difficult to execute methods remotely that uses local devices or re-executions hamper the actual results.

1) MAUI: Cuervo et al. [13] proposed a fine grained code offloading system for saving energy of mobile devices in making smartphones last longer with code offload called MAUI. MAUI introduces both static and dynamic decision of method level partitioning. Initially the developers annotated the remote executable methods as remoteable and locally executable methods as local. The proposed system automatically identified the remoteable and non-remoteable methods, and then automatically performed migration on application methods.

The MAUI network, device and program profiler data were used by MAUI solver as input to the optimization solver in order to identify local and remote executable methods. The goal of the optimization solver was to save device energy with respect to different latency. The MAUI system solved the following 0-1 integer linear programming using indicator variable I_v . For local executable method v the value of I_v is 0 otherwise 1. E_v^l and E_v^r represents energy requirement for method v in locally and remotely respectively and T_v^l and T_v^r are time required to execute the method v in locally and remotely respectively.The parameters $B_{u,v}$ and $C_{u,v}$ represent necessary data size and energy cost for transferring states from u to v.

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

such that: $\sum_{v \in V}((1 - I_v) X T_v^l) + (I_v X T_v^r))$+ $\sum_{(u,v) \in E}(|I_u - I_v| X B_{u,v}) \leq L$

and $I_v \leq r_v, \forall_v \in V$

The goal of the objective function is to maximize the energy saving by transferring tasks remotely. The first constraint guarantees that execution time for a program must be within total execution time L, and second constraint guarantees that methods that are annotated remotely can be executed remotely. MAUI significantly increases the energy saving of

mobile devices by considering the user mobility and network dynamics. The MAUI system behaves incorrectly if developers make mistakes to label a method. The system requires expert programmers to annotate methods for local and remote execution. The system is not scalable with increasing number of workloads and used serial execution of offloaded tasks. The proposed system is evaluated on the face recognition, gaming and voice based language translation applications.

2) ThinkAir: Kosta et al. [14] proposed Thinkair that enables the parallelization of method execution using multiple Virtual Machine (VM) images for the enhancement of performance and battery lifetime of mobile devices. It introduces method level partitioning for code offloading and exploits the concept of smartphone virtualization in the cloud. The proposed system focuses on the elasticity and scalability of the cloud which increases the power of mobile cloud computing by parallelizing method execution using different VMs.
ThinkAir provides an efficient environment to perform on-demand resource allocation for tasks and supports parallelism by dynamically creating, resuming and destroying VMs in the server end whenever necessary without affecting the performance of the application. The system is evaluated on the N-queen puzzle, face detection, virus scan, and image merger applications. The limited factor is that migrated thread blocked until the offloaded thread returns which reduces the concurrency of their work.

3) Cuckoo: Kemp et al. [15] proposed an offloading framework for Android named Cuckoo, including a runtime system, a resource manager application for mobile device user and a programming model for developers. Herein, the runtime system dynamically identifies which methods are executed locally and which are remotely. The resource manager stores the remote resource information. The programming model supports local and remote execution, bundles the local and remote code in a single package. The model helps to discover remote resources for application execution including laptops, home servers and cloud resources.
In Cuckoo, additional communication is required because same code execute both locally and remotely that consumes energy and increases the execution time. The Cuckoo system intercepts all method calls and only check the accessibility of the remote resources for selecting offloadable method which is not enough.

4) Offloadable Apps using SmartDiet: SmartDiet [16] tasks are to identify the limiting factors that reduce offloading opportunities. It is used to calculate the energy-saving potential of offloading communication-related tasks. The system uses method level application execution partitioning framework called ThinkAir for partitioning the application. There are many constraints that limit the remote execution of offloadable method which makes it difficult to manually identify. SmartDiet is the proposed toolkit for identifying such constraints automatically. The main purpose of the paper is to provide suggestion of code modifications for removing the identifiable constraints. The system is used for evaluating the energy-efficiency and performance of code reconstruction at the stage of development. Getting the difficulties from SmartDiet, it provides a guidance to the programmers to improve application implementation for better energy efficiency.
The authors in [17] uses SmartDiet [20] for code offloading in order to increase the battery life of mobile devices. This paper is the study of feasibility of method level offloading in network intensive applications. It uses an open source Twitter client to exemplify the associated issues. The paper uses ThinkAir framework for offloading but disable all the dynamic decision making feature. The limiting factor of the paper is to dependent on expert developer or domain expert for methods annotation.

5) Mobile code offloading from concept to practice and beyond.: In [18], the paper addresses that code offloading technique faces many challenges in practical usage and adapt a generic code offloading architecture systematic approach. The authors also identify the key limitations for code offloading and provide the solutions for these limitations based on the theoretical and experimental analysis. Based on the solutions, the paper presents and evaluates use cases which give insights in code offloading. The proposed solutions increase the performance of mobile application without producing extra overhead in the devices. It also reduces the amount of data transfer between device and cloud server. This paper identifies inaccurate code profiling, integration complexity, dynamic configuration and offloading scalability as major challenges and technical problems in code offloading

technique.
In the proposed system, handling multiple offloading requests, server creation can be time consuming and costly. The system is evaluated on the face recognition, gaming, chess game, and puzzle application.

B. Thread level granularity

For supporting the scalability with increasing workload or proper resource usage, thread level granularity is used for partitioning an application. In thread level granularity, thread is migrated to the remote server for execution. The remote server execute multiple independent thread parallelly which makes the system scalable.

1) CloneCloud: Chun et al. [19] created CloneCloud which have enabled mobile application to migrate thread to device clones operating in a computational cloud. The system employs dynamic profiling and static analysis to partition the mobile application in order to optimize overall execution cost. Herein, the application is partitioned, send it to the clone, execute the code and re-integrating the migrated thread back to the mobile device. The selection of local and remote execution of methods is taken by the optimization solver. The purpose of the optimization solver is to deliver optimal application methods to the cloud from the mobile devices.
The system has static analyzer for code offloading which require experienced domain experts. This paper consider limited environmental conditions and assume resources that are not available on the cloud. The CloneCloud system is evaluated on the virus scanning, image processing, and privacy preserving targeted advertising applications.

2) COMET: Gordon et al. [20] propose code offload by migrating execution transparently called COMET, a multi-threaded application that can be migrated freely between devices depending on the workload and use distributed shared memory for offloading. The aim of the system design is to require only program binary or no manual effort, execute multi-threaded programs correctly, improve speed of computation, resist network, server failures and generalize well with existing applications. The paper proposes a scheduling algorithm which can give some loose guarantees on worst case performance.
The system requires enough information in order to restart the computation of remote devices if any type of failure occurs. The COMET framework focuses on performance improvement and energy saving for image editor, turn based games, a trip planner and math tools application.

3) EMCO: Flores et al. [21] proposed EMCO, where high computation required operations are identified and partitioned at code level and offloaded for remote processing. Mobile device consume services from different cloud by efficiently utilizing solutions in their delegation model. The authors proposed a fuzzy logic engine which considered both mobile and cloud variables like performance metrics, parallelization of tasks, elasticity etc.Rules are introduced asynchronously to the mobile device using notification services. The paper also proposed evidence based learning methods to enrich the offloading decision and implement a prototype for fuzzy logic engine.
In code offloading, components of an application can be marked as remote executable by a developer or an automated mechanism. The purpose of the fuzzy logic engine is to decide which components of an application is offloaded or not. The logic engine profiles bandwidth, connectivity, size of computational data, and applies certain logic over them to make advantage by code offloading.

C. Data Size Level Granularity

Data level granularity refers to deliver the data size of a task to the remote server to execute the application parallely. Data stream deliver to the remote server either serially or parallely. In parallel execution, the remote executable tasks increase the performance and reduce battery usage.

1) Odessa: What computation to offload and how to structure the parallelism across the mobile devices and cloud, Ra et al. [22] proposed Odessa which dynamically makes offloading and data parallelism decisions for mobile interactive perception applications. As performance depends on scene complexity and environmental factors like network and device capabilities, this paper found that offloading and parallelism choices should be dynamic for an application.
Odessa considered a variety of execution environment, network, device, application inputs and use networked computing infrastructure to enhance the capabilities of mobile devices. The proposed system evaluated face recognition, object, pose recognition and gesture recognition. The proposed system showed improvement in performance over partitioning by domain

expert but domain expert definition was not defined or verified.

2) COSMOS: Shi et al. [23] proposed Computation Offloading as a Service for Mobile Devices called COSMOS. The paper provides computation offloading as a service for solving the problem of mobile device computing resources demand.It also introduces how the cloud provider offers the resources to the device user for reducing cost and performance improvement. The system manages cloud resources for offloading requests for reducing monetary cost to the cloud provider.

The proposed system only considerd network connectivity context variable for offloading decision. There was no pricing model defined for the cost. The system was evaluated on face recognition, voice recognition and chess game application.

IV. COMPARISON OF CODE OFFLOADING MECHANISMS

A comparison of the existing mechanisms for code offloading may address the way to point out to the new solution for code offloading.

Level of Execution: In mobile applications, tasks are either executed to the mobile devices or delegated to the remote cloud servers or cloudlet for remote execution. The metric describes how to execute the partitioned code to the cloud and is it serial or parallel depending on the different variable parameters.

Level of Profiling: In code offloading, tasks are delivered to the remote resources where tasks can be method, class, thread or data stream. This metric describes which parts method, thread or data stream of an application are offloaded to the cloud.

Security Requirements: In order to protect executable code while delivering or storing to the remote cloud from attacks. Different types of security such as encryption,authentication can be assigned to the application or remote server are needed to be secure. The metric describes how much the server secure for the executable code.

Code Profiler: The tasks of an application can be partitioned either statically by expert developer or identify remote executable tasks dynamically at runtime. The metric describes how to partition the code manually or automatically.

Offloading Adaptation Context: What to be offloaded, when to offload, how to offload or where to offload are the main adaptation contexts for code offloading. The metric describes which adaptation contexts are used for taking offloading decision.

In TABLE 1, we have summarised the comparison of existing code offloading mechanisms based on these performance metrics. From the research, we have found that dynamic method level parallel execution to the secure remote clouds increases both performance and battery lifetime. MAUI and CloneCloud execute offloadable codes to remote clouds serially and the rest of the papers execute parallely. MAUI, ThinkAir, Cuckoo, SmartDiet and MCO implemented method level code partitioning. On the other hand, Odessa and COSMOS impemented data size level and the rest of the papers implemented thread level partitioning. Among the papers surveyed, only EMCO and MCO considered all the apaptation concext (i.e., what, when, where and how to offload) of code offloading. EMCO considered dynamic thread level parallel execution and MCO considered dynamic method level parallel execution. The discussion and findings for code offloading are described in the next section.

V. DISCUSSION AND FINDINGS

This section describes the discussion and findings which help the researchers to do research in the area of code offloading. Throughout this paper, many research works have been analytically related to the code offloading and a comprehensive survey of all the studies would be impossible. The referenced research is selected based on our knowledge of the topics. From the research, we have found different findings which help researchers to take necessary decisions for building code offloading framework.

In code offloading, it is necessary to identify which methods are executed locally and which are executed remotely called partitioning. Detecting the methods or threads which are executed either locally or remotely during runtime is a challenging task and is an optimization problem. There are three types of partitioning for denoting the methods, threads or data size which can be partitioned either statically or dynamically. In code offloading, local and offloadable codes should be annotated so that it is possible to identify the local or remote executable tasks at runtime. From the analysis, it is found that about 55% of the work are

Comparison Metrics					
Name of Paper	Level of Execution	Level of Profiling	Security Requirements	Code Profiler	Offloading Adaptation Context
MAUI	Serial	method	No	Manual	What, When
CloneCloud	Serial	thread	No	Automated	What,When
ThinkAir	Parallel	method	No(Future Work)	Manual	What,When
Odessa	Parallel	Data Size	No	Automated	What, When
COMET	Parallel	thread	No	Automated	Little What,How
Cuckoo	Parallel	method	No(Future Work)	Manual	What, how
EMCO	Parallel	thread	No	Automated	What,When,how, where
SmartDiet	Parallel	method	No	Manual	What,when
OffloadAble Apps using SmartDiet	Parallel	method	No	Manual	What,when
COSMOS	Parallel	Data Size	No	Automated	Where, How
Mobile code offloading: from concept to practice and beyond	Parallel	method	No	Automated	what, when, how where

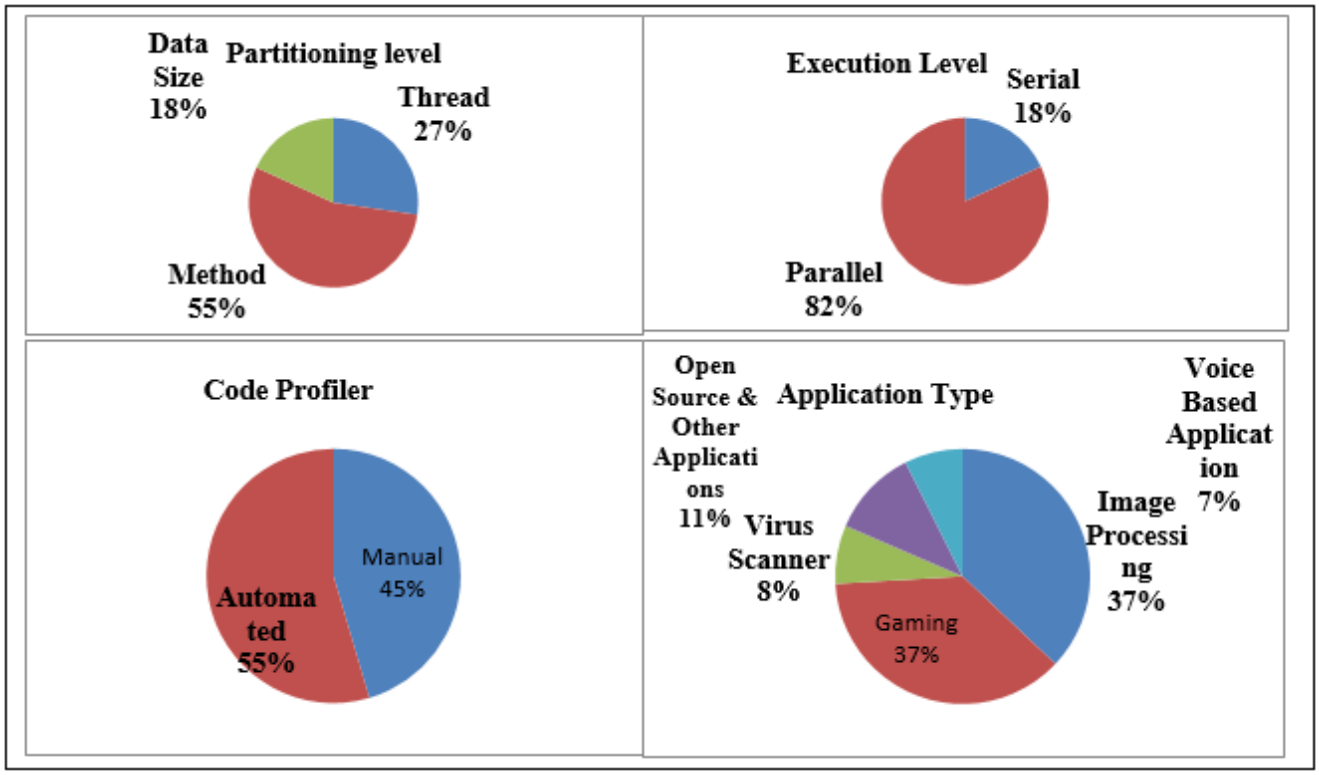

Fig. 6: Findings on Code Offloading

using automatic code partitioning and 45% are using static code partitioning. The partitioning of the code offloading can be method level, thread level or data size level. From or findings, about 55%, 27% and 18% research works use method, thread and data size level code partitioning respectively. The execution of the offloadable tasks could be executed remotely either parallely or serially. About 82% of the works support parallel and 18% works use serial execution of remote tasks. About 37%, 37%, 8%, 11%, and 7% works used image processing, gaming, virus scanner, open source and other applications and voice based applications respectively. The entire findings are displayed in Fig. 6.

The researchers in code offloading could take decisions while building code offloading mechanisms. Most research works used parallel method level dynamic code offloading mechanism and evaluated their work on image processing and gaming applications.

VI. OPEN CHALLENGES

This section describes the major challenges that hinder the performance of computation offloading in mobile cloud computing. The open challenges assist the researchers to find new research directions in the domain of code offloading.

1) Combination of Serial and Parallel Execution of the Offloadable Code: The offloadable part of an application that executed to the cloud can be serial or

parallel. The serial execution of the application to the cloud is time consuming and devices needs to wait for future execution. The parallel execution of the application requires synchronization among offloadable parts of the application. In order to remove the limitation of serial and parallel execution of application separately, the combination of the serial and parallel execution of application can be applied to increase the performance and battery lifetime. It will be an optimization problem to identify which parts are to be executed serially and which parts in parallel based on the network connectivity, server load.

2) Cloud Provider Execution Cost Based on Offloaded Task Not on Hour Basis : Cloud computing makes it possible for the users to pay on demand basis or on the usage of resources. Sometimes the cloud providers allocated resources that are not fully utilized, but the user needs to pay for the resources. The cloud provider could be charged according to the execution time of the offload able task for the proper use of the resources. The pricing of the offloadable part of the applications are based on the execution time, size of the executable part, frequency of execution. The client provider needs to discard the cost or time for creating, resuming or destroying VM. As a result, cloud providers and mobile device users are both in the win-win situation by charging only for computation offloading and avoiding wasting of cloud resources.

3) Data Protection from Attacks: It is necessary to protect data while delivering or storing it to the remote cloud servers. Herein, Security and privacy are one of the important factors that hinder the successful seamless execution of an application to the cloud via the internet. In code offloading, sensitive and confidential data is delivered to the remote server for execution which is at risk from insider attacks. Data center stores multiple clients data to a single server which sometimes make it harder to secure one user data from others. A number of code offloading mechanisms analyze the need of security and privacy but few of those have properly implemented the security to their framework.

4) Seamless Communication among Remote Clouds: The offloadable codes are delivered to the remote cloud for increasing performance and reducing energy consumption. The work assigned to the remote cloud server will be increased by another clients request having heavy computation. The remote server will not perform correctly, if distance between the cloud server and client are increasing. As a result, the server needs to send the assigned work back to the client. The problem can be mitigated by sending the latest computing task of the server to other servers which are optimal called live VM migration.

VII. Conclusion

In this paper, we have surveyed the state of the art code offloading techniques for mobile systems and examine how mobile device users and cloud computing make the computation offloading feasible. Different type of metrics for code offloading has been presented to better analyze offloading mechanisms. The paper has also identified the types of applications that are the candidate for code offloading and demonstrated the findings gathered from the background. Finally, we describe the open challenges with a view to assist the researchers in finding new research direction in the field of code offloading.

References

[1] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 369–392, 2014.

[2] E. Ahmed, A. Gani, M. Sookhak, S. H. Ab Hamid, and F. Xia, "Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges," *Journal of Network and Computer Applications*, vol. 52, pp. 52–68, 2015.

[3] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.-I. Yang, "The case for cyber foraging," in *Proceedings of the 10th workshop on ACM SIGOPS European workshop*. ACM, 2002, pp. 87–92.

[4] R. K. Balan, M. Satyanarayanan, S. Y. Park, and T. Okoshi, "Tactics-based remote execution for mobile computing," in *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM, 2003, pp. 273–286.

[5] A. J. Oliner, A. P. Iyer, I. Stoica, E. Lagerspetz, and S. Tarkoma, "Carat: Collaborative energy diagnosis for mobile devices," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2013, p. 10.

[6] G. Huerta-Canepa and D. Lee, "An adaptable application offloading scheme based on application behavior," in *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on*. IEEE, 2008, pp. 387–392.

[7] 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*. IEEE, 2006, pp. 10–pp.

[8] B. Seshasayee, R. Nathuji, and K. Schwan, "Energy-aware mobile service overlays: Cooperative dynamic power management in distributed mobile systems," in *Autonomic Computing, 2007. ICAC'07. Fourth International Conference on*. IEEE, 2007, pp. 6–6.

[9] G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and R. Chandramouli, "Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 15, no. 9, pp. 795–809, 2004.

[10] D. Shivarudrappa, M. Chen, and S. Bharadwaj, "Cofa: Automatic and dynamic code offload for android," *University of Colorado, Boulde*, 2011.

[11] H. Qian and D. Andresen, "Jade: Reducing energy consumption of android app," *the International Journal of Networked and Distributed Computing (IJNDC), Atlantis press*, vol. 3, no. 3, pp. 150–158, 2015.

[12] P. B. Costa, P. A. Rego, L. S. Rocha, F. A. Trinta, and J. N. de Souza, "Mpos: a multiplatform offloading system," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. ACM, 2015, pp. 577–584.

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

[14] 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*. IEEE, 2012, pp. 945–953.

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

[16] A. Saarinen, M. Siekkinen, Y. Xiao, J. K. Nurminen, M. Kemppainen, and P. Hui, "Smartdiet: offloading popular apps to save energy," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 297–298, 2012.

[17] ——, "Offloadable apps using smartdiet: Towards an analysis toolkit for mobile application developers," *arXiv preprint arXiv:1111.3806*, 2011.

[18] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: from concept to practice and beyond," *Communications Magazine, IEEE*, vol. 53, no. 3, pp. 80–88, 2015.

[19] 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*. ACM, 2011, pp. 301–314.

[20] M. S. Gordon, D. A. Jamshidi, S. A. Mahlke, Z. M. Mao, and X. Chen, "Comet: Code offload by migrating execution transparently." in *OSDI*, 2012, pp. 93–106.

[21] H. Flores and S. Srirama, "Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning," in *Proceeding of the fourth ACM workshop on Mobile cloud computing and services*. ACM, 2013, pp. 9–16.

[22] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: enabling interactive perception applications on mobile devices," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pp. 43–56.

[23] C. Shi, K. Habak, P. Pandurangan, M. Ammar, M. Naik, and E. Zegura, "Cosmos: computation offloading as a service for mobile devices," in *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2014, pp. 287–296.

Md. Irfan received his B.Sc. and M.Sc. in Software Engineering, Institute of Information Technology, University of Dhaka in 2014 and 2016 respectively. He is now working as a lecturer in the Department of Computer Science and Engineering, Barisal University. He has research interest in Software engineering and mobile-cloud computing.

Bidoura Ahmed Hridita received his B.Sc. and M.Sc. in Software Engineering, Institute of Information Technology, University of Dhaka in 2014 and 2016 respectively. She has been doing research in the area of Cloud Computing.

Md. Shariful Islam received his BS and MS in Computer Science from the University of Dhaka, Bangladesh. He obtained his PhD degree in Wireless Networking from the Department of the Computer Engineering, School of Electronics and Information, Kyung Hee University, South Korea in 2011. He is now working as a Professor in the Institute of Information Technology (IIT), University of Dhaka, Bangladesh. He has been teaching a good number of courses related to Computer Networks, Wireless and Mobile Systems, Security, Information Technology Project Management, etc. to graduate and undergraduate students of reputed universities. He has research interests in wireless networking, Wireless Mesh Networks, Information security, Cloud computing, etc. He has published a good number of research papers in international conferences and journals.

Mohammad Shoyaib received his MS degree in computer science from the University of Dhaka, Bangladesh, in 2000 and in 2012, he has completed his PhD degree from the Kyung Hee University, South Korea. His research interests include pattern recognition and machine learning.