DOI: http://dx.doi.org/10.26483/ijarcs.v9i2.5767

ISSN No. 0976-5697

Volume 9, No. 2, March-April 2018

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

REVIEW ON COMPUTE AUGMENTATION TECHNIQUES IN MOBILE CLOUD COMPUTING

V. Suganya
Ph.D. Research Scholar,
Dept. of Computer Science and Applications,
SCSVMV, Kanchipuram – 631 561,
Tamilnadu, India

Dr. M. Kannan
Asst. Professor,
Dept. of Computer Science and Applications,
SCSVMV, Kanchipuram – 631 561,
Tamilnadu, India

Abstract: Nowadays, Smartphones has become one of the gadgets of all peoples due to its technological advancements which led to fast development of resource-intensive applications for watching videos, web surfing, interactive gaming, augmented reality and location tracking etc., Since the resource-poor mobile devices have only limited CPU, RAM, and battery, it is not capable to execute the resource-hungry applications. To overcome this impediment, the Computation/ Code offloading method of Mobile Cloud Computing (MCC) can be used. MCC is an infrastructure where the cloud server resources can be augmented by mobile devices. In Computation offloading, the compute-intensive part of the application is offloaded to the resource-rich cloud servers and returns the result back to the smartphone. This paper analyzes and surveys the types of offloading processes, application partitioning, different kinds of frameworks and challenges in offloading the computation part from mobile to cloud.

Keywords: Mobile Cloud Computing, Computation Offloading, Application Partitioning, Augmentation techniques, Offloading Frameworks

1. INTRODUCTION

In recent years, every people in each corner of the world are using mobile devices such as smartphones, smartwatches, tablets, and notebooks etc., In earlier days mobile devices are used only for voice communication but nowadays, mobile devices are advanced in terms of processing speed, sharper screen and more sensors to run content rich or computation-intensive applications such as augmented reality, face recognition, interactive games, online video streaming, object tracking, web surfing, speech and object recognition. Though the mobile has been steadily improving, it is not capable to execute the compute-intensive applications for the following reasons: a) Shorter battery lifetime, b) Limited Processing speed and c) Low Memory capacity. To increase the response time and reduce energy consumption an alternative solution is needed.

Cloud computing is getting popular due to its features such as elasticity, scalability, low cost and so on. The response time of applications executed in the mobile devices can be increased by utilizing the unbounded compute resources of the cloud servers can be purchased on demand in any quantity at any time. E.g. Amazon EC2 compute. Since the mobile devices have only limited storage, the provisioning of data centers by cloud servers can extend the memory capacity of the mobile devices. E.g. Amazon S3 storage. Features like scalability, dynamic resource allocation, and quick provisioning make cloud the best partner for mobile computing.

The key contribution of this paper is to analyze the techniques, works used on computation offloading in mobile computing. Various papers are received to find the research gap in existing computation offloading techniques which will be useful for upcoming researchers to find novel solutions for the same.

The rest of the paper is organized into following sections. Section 2 discusses the definition, architecture and applications of MCC. Section 3 describes the offloading process and its types. Section 4 presents the requirements for the offloading process. Section 5 highlights the challenging impediments in offloading. Section 6 describes and analyses various offloading works. Section 7 concludes the paper. Section 8 suggests the future research directions.

2. MOBILE CLOUD COMPUTING (MCC)

A. MCC Architecture

Mobile Cloud Computing (MCC) combines the strength of Cloud computing with Mobile terminals. So that the resources hosted by cloud servers can be augmented by mobile devices. It's an infrastructure where both data storage and processing happen outside of mobile devices. Mobile users can access the application, data, and cloud services from anywhere at any time through the internet. Mobile cloud computing can be done in three ways. First, a group of mobile devices can form a cloud by sharing their storage and processors. Second, the mobile user makes use of resource-rich cloud resources such as virtual instances, server, storage, load balancing, network resources, and databases. Third, mobile devices in the local vicinity connected to the servers with limited computation and storage in the hotspot and referred to as cloudlets.

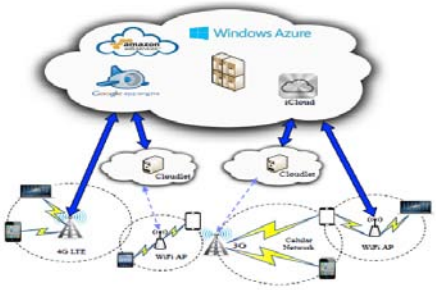

Figure 1. Mobile Cloud Computing Architecture [1]

The MCC architecture consists of three tiers of which tier 1 is Master cloud, tier 2 is the Local cloud or Cloudlet and tier 3 is Mobile device cloud. Master cloud is the cloud server with large resource pool and high computation power which can be availed to the user at any time. Users located far away from the master cloud connect to the local cloud or cloudlet via a wireless access point to decrease latency and lower battery consumption. A cloudlet is a trusted cluster of resource-rich computers which is available to the nearby mobile device via WLAN hotspot. The third tier is the mobile device cloud, which is formed by the mobile devices in the nearby proximity via Bluetooth by sharing its storage and computation power. It is used when an internet connection is not available.

B. MCC Benefits

The Mobile Cloud Computing has the following benefits: Extended battery life by means of mobile computational offloading, Data storage from unlimited virtual cloud, Increased processing power by allowing resource-intensive applications to run on the cloud servers, Dynamic provisioning of resources without advanced reservation, Scalability to meet all unpredicted requirements of a mobile user, Reliability since data stored in cloud are secure and Ease of integration because cloud providers could integrate multiple services into one.

C. MCC Applications

The Mobile Cloud Computing applications include Mobile Commerce, Mobile Sensing, Multimedia sharing, Mobile Learning, Mobile Healthcare, Mobile Gaming, Mobile Social Networking, Crowdsourcing, Collective sensing, Location-based mobile cloud service and augmented reality. [2]

3. COMPUTATION OFFLOADING

A. Offloading

In MCC, Computation offloading from mobile devices to the cloud is a method to reduce execution time and extend the battery life of mobile devices by transferring and executing the mobile application outside of it.

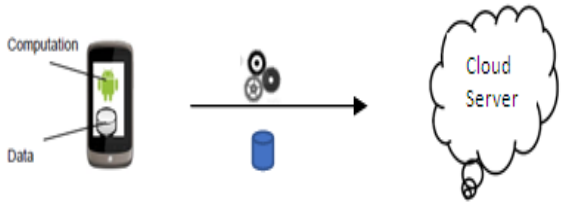

Figure 2. Offloading to Cloud

In the above figure, both the computation and data of the mobile device are offloaded to the resource-rich cloud server for execution and storage which returns the result back to the mobile device. Thus, the resource-poor mobile device executes the task with high response time and lower battery
consumption.

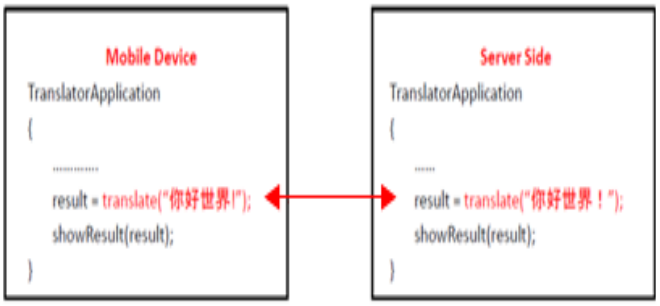

Figure 3. Computation Offloading in Translator Application [3]

In the above code snippet when the translate method encounters, the execution of the mobile end stops and transfers the data and control to the cloud server for processing. The cloud server processes it and returns the result back to the mobile device. After receiving the result, the mobile device starts resuming the execution with further lines of code in the translate application.

Types of Offloading

Offloading the computation of an application to the cloud can be done in two ways. They are:

i) Virtual Machine based cloning

In VM based cloning, the entire phone image is cloned in the virtual machine of the cloud server. The cloud server then loads that appropriate virtual machine and executes the computation. This approach requires a lot of data transfer for Synchronization in the VM thereby reduces the performance.

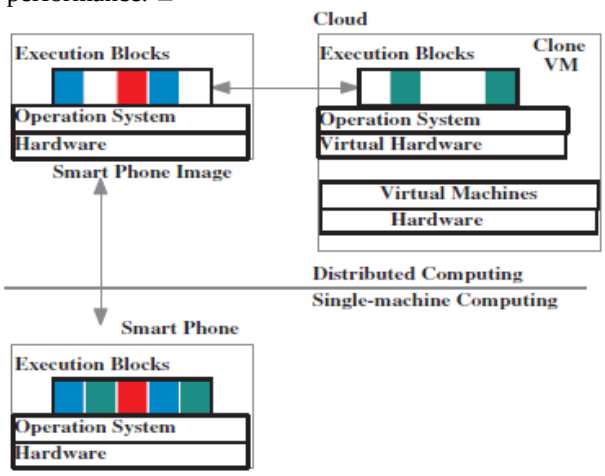

Figure 4. Clone Cloud Framework [4]

ii) Task Partitioning

Here the application code is partitioned and the offloadable component is outsourced to resource-rich cloud servers. There is no need for synchronization in task partitioning approach thereby provides high response time with reduced battery consumption. The example framework for this approach is MAUI:

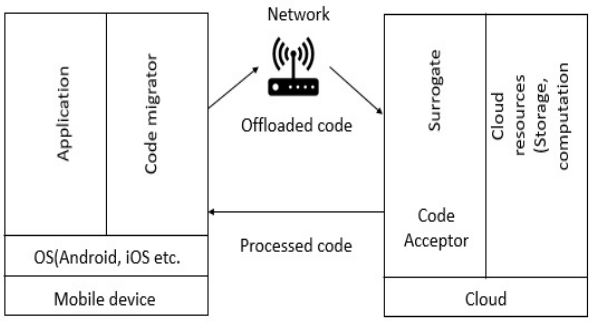

Figure 5. Application Partitioning

4. OFFLOADING REQUIREMENTS

A. Profiling

To make correct offloading decisions with low overhead, information about the device (energy consumption, screen, and CPU utilization), application (execution time, required resources) and network characteristics (bandwidth, latency and packet loss) should be collected and such process is called as profiling. After profiling, the solver makes use of this profiling metrics and takes the decision to whether offload or not.

i) Device Profiler

There are two ways to measure the energy consumption, (1) Software and (2) Hardware monitor. Software called Power Tutor is used to measure the power consumption of the application and device. A hardware called Monsoon Power Monitor is attached to the smartphone battery by supplying power to measure the power consumption when the computation is transmitted from mobile to the resource-rich cloud server. However, comparatively the hardware monitor is better than the software monitor in measurement.

ii) Application Profiler

It collects the application characteristics such as parameter data size (send size, receive size, and transfer size), offloaded code execution time, CPU utilization time, the number of instructions executed, the number of calls, the dependency between the calls and memory allocation size either statically or dynamically.

iii) Network Profiler

It collects information about network environment or connection statuses such as Wi-Fi/3G/4G/Wi-Max connectivity, available bandwidth, Round Trip Time (RTT) by sending and receiving packets to the VMs on the cloud and to measure the number of packets transmitted and received per second, Uplink and Downlink data rate and Signal strength.

B. Application partitioning

Application partitioning is the pre-processing step for offloading the computation in mobile computation offloading frameworks. Computation offloading makes use of partitioning to segregate the compute intensive logic into different partitions, so that it can be executed in a distributed environment. Splitting the applications in to several parts and also preserving the semantics of the application genuineness. The application partitioning consists of various levels of granularity for partitioning the compute intensive tasks. They are module level, method level, object level, thread level, class level, task level, component level (group of classes), bundle level (groups of java class of applications), allocation-site level (all objects in the specific site as a unit) and hybrid level (partitions of different granularity). The objective of partitioning is to (i) improve and enhance the performance, (iii) to execute on multiple remote servers, (iv) to solve the prolonged battery and memory constraints on mobile devices, (v) to reduce the network overhead and (vi) to reduce the burden of the developers.[5]

C. Offloading Decision

Decision made at the right time and in the right way to will leads to achieve greater benefits in offloading the computation intensive tasks. The decision engine in the computation offloading framework the following four factors (What, Where, When and How).

i) What to Offload?

A resource hungry mobile application is decomposed into set of fine-grained task components. But not all these components will be offloaded to the resource rich cloud end due to the occurrence of communication delay or the task itself cannot offloaded since it may access local device parameters such as sensors, user interface, camera etc., Therefore the components in the application should be categorized in two ways (1) Offloadable and (2) Non-offloadable using any optimal application partitioning strategy it is necessary to determine which subset of tasks is worth to be offloaded to resource powerful cloud server and what has to be executed locally by estimating time and cost incurred.

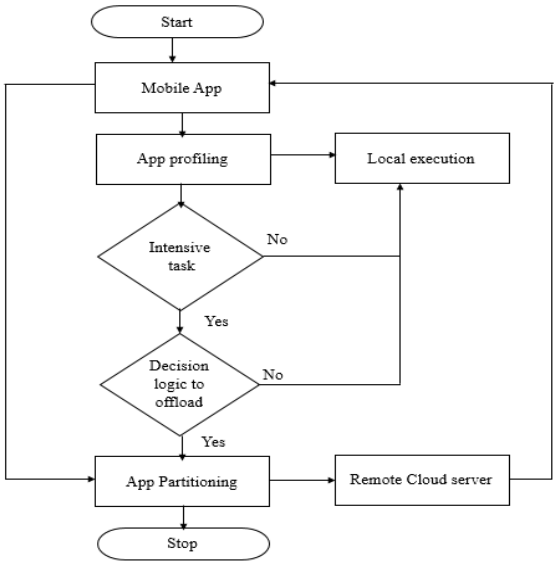

Figure 6. Application Partitioning Flowchart

ii) When to Offload?

It is necessary to determine when to offload a compute intensive task to resource rich cloud end by estimating whether the total execution time of the component in the mobile device is greater than both the total execution time in cloud server and transfer time of the component mobile to cloud and vice versa. Offloading will be beneficial if all the network metrics are favourable with less communication cost.

iii) Where to Offload?

There are multiple offloading destinations available to deploy the offloading application components such as local mobile clouds via Bluetooth, cloudlets via Wi-Fi and high-end cloud servers through cellular networks. The determination of the destination is chosen by offloading simple computational task to less resource computing targets without any internet connection, heavy computation tasks and large datasets to resource rich cloud servers with high network bandwidth and in case of high response time, the computation will be offloaded to nearby by cloudlets with enough resources and WLAN hotspot. For these cases, code offloading decision is made to find where to offload by determining suitable cloud resources based on the wireless network available.

iv) How to Offload?

Though an optimal decision has made using various parameters, if the wireless channel link is weak, it leads to high energy consumption and communication delay. So, to run the offloading process optimally, it is necessary to choose among different available wireless channels for offloading the compute intensive tasks by estimating parameters such as energy cost, link speed, availability, link quality, monetary cost, and conjunction level of channel using an effective strategy to achieve the optimal objective.

5. CHALLENGES IN OFFLOADING

Though mobile computation offloading has advantages such as increased computing power, storage capacity, and enhanced battery power, it has the following challenges:

A. Security and Privacy

Security and privacy are the main issues in mobile computation offloading. The security issues must be resolved in three scenarios: mobile device, cloud servers, and data during transmission over the communication interface. The security and privacy issues of mobile computation offloading are divided into three levels: mobile device, mobile network or wireless communication channel, and the cloud infrastructure.

B. Mobile Device Security issues

The security issues in the mobile device are: information stealing malware, spam, phishing, data loss from lost or stolen devices, data leakage from the poorly written application, vulnerabilities in hardware or OS, unsecured Bluetooth or Wi-Fi. Eg: information stealing malware, fake websites, digital wallet hacking, unwanted messages etc.,

To prevent the mobile device from the above said issues, it is necessary to i) update the OS periodically, ii) third-party application should not be installed, iii) should exchanging data from strange mobiles, iv) suspicious and unexplained links should not be tried, v) the Bluetooth, Wi-Fi interface should be in off mode when not in use, vi) remote data wiping technique should be used when the device is stolen or lost.

C. Communication channels security issues

Issues related to wireless communication channel are Access control attacks, Confidentiality attacks, Integrity attacks, Authentication attacks and availability attacks. Eg: MAC spoofing, WEP cracking, Man-In-The-Middle attack, Denial of service, AP Phishing, VPN login cracking, Beacon flood, Frame injection, Reply attacks etc., To mitigate these communication channel security issues, the following approach has to be followed: 1) The encrypted data should be used while transmission, 2) secure transmission protocols such as Https and SSL should be used for data transfer, 3) to avoid MITM attacks, public key encryption technique is better, 4) socket programming, strong password, and biometric authentication will enhance the security of data while transmission, 5) should avoid using public access points, and c) the Wi-Fi and Bluetooth interface should be switched off when the mobile is not in use.

D. Cloud infrastructure security issues

The security issues in the cloud platform are Integrity, Digital rights management, Virtual machine attacks, and platform level attacks. Eg: Data and application integrity, pirated distribution of digital contents, side channel attacks, SQL injection etc., To protect the data in the cloud from the above said issues, the following techniques has to be used: 1) security technologies such as Virtual private network, access control, encryption can be used, 2) automatic recoverable of user's data is preferable during data lost or erased by an attacker, 3) secure key management technique can be used, 4) to maintain the privacy and security of data, the user should be provided with the location of the data offloaded and 5) implicit authentication techniques will reduce the risk of fraud.

To prevent the security breaches, the offloaded data must ensure the following most generic criteria:

Authentication is a verification process. In Computation offloading, it is a bi-directional verification. i.e., the user who requests to access the resource on the cloud should be an authenticated one to claim it and the cloud which is servicing the request to the user should be an authenticated cloud server to provide the service.

Authorizing an application to access user data without releasing the user's credential can save energy of mobile devices and reduce data transportation overhead.

Data Integrity means data stored/offloaded code by the user at remote storage in the cloud is not modified by the cloud at any cost. To ensure this criterion, the cloud server must secure and protect themselves from malicious attacks.

E. Platform diversity

Mobile devices use the different architecture and operating systems when compared to cloud servers. Not all the offloading framework is suitable for deployment on all mobile platforms. So, it is necessary to develop a framework which should support all mobile devices and deploy on the cloud end regardless of the platform and hardware.

F. No continuous connectivity

Mobile users usually move from one location to other. On the fly, the connection may be lost which leads loss in packet transmission. To ensure the successful execution of offloaded application component, the offloading approach should be equipped with fault tolerance mechanism to retransmit the lost packets and to minimize the power and response time.

G. Low Scalability

Scalability is the process of allocating computing resources to the available resource-rich virtual machines and to maintain the overhead of the load when it exceeds the availability. To ensure the user's quality of experience, an optimal resource allocation strategy is needed to allocate the tasks to the server with minimum response time.

H. Bandwidth scarcity

One of the factors that affect offloaded task is bandwidth. With the availability of low bandwidth, the offloaded processing time will be delayed which leads to low latency.

I. Cloud Pricing

Cloud pricing is applied to cloud computing. When an intensive task encounters in mobile, offload it to the cloud servers for execution. But the cloud service providers will provide the service for pricing in return. The more the services, the more pricing will be.[6]

6. RELATED WORKS

Mobile computation offloading framework consists of various Application programming interfaces, Partitioning algorithms, Decision engine, Profiler, compilers, Security mechanism etc., to offload the compute-intensive task to the high-end cloud servers optimally. Each existing framework will use the different approach to offload the intensive task to the remote cloud servers. [6] In this section, some of the prominent existing offloading works are discussed.

Chun et al., [4] proposed Clone Cloud framework which offloads the unmodified mobile applications executable without any developer intervention for execution from the mobile device onto the virtual clones running in the resource-rich cloud infrastructure. The Clone cloud's application partitioner automatically finds out the partitions through static analysis and dynamic profiling at a fine granularity level to reduce the execution time and energy for the specific computation and communication environment without any source annotations and programmers involvement, the optimization solver picks the appropriate application methods to transfer the computation to the VM clones and receives the result by minimizing the application partitioning cost. Migration in Clone Cloud is effective in terms of the following (a) per-process migrator thread which suspends, package, resume and merge thread state for the specific process (b) a per-node node manager is used for node-node communication of thread package, synchronization of clone images and (c) partition database to identify the appropriate partition to offload. Clone Cloud prototype delivers the execution of offloaded application speed and energy reduction rate at 20x speed. The main drawback of this approach is synchronization of clone image which leads to performance overhead.

Cuervo et al., [7] proposes a MAUI system that enables the fine-grained energy-aware computation offloading to cloud infrastructure. MAUI framework offers semi-automatic offloading of code i.e., there is no need for the developer to code the logic for shipping the computation instead the developer should annotate only the functions which are to be offloaded and the optimization engine at runtime will then decide which intensive annotations can be offloaded by considering all profiling (device, program and network) information with necessary state thereby reducing the burden of the programmer. It uses 0-1 integer linear programming for solving the optimization problem. MAUI is based on .Net framework for method level code offloading. MAUI does not support the cloud's scalability feature and the adaptability of mobile application for distinct devices.

Kosta et al., [8] presented Think Air framework which migrates the application to the cloud exploiting the smartphone virtualization concept in cloud with method-level code offloading. Its goal is to enhance the battery power of smartphone. It overcomes the lack of MAUI's scalability by parallelizing the execution of method in multiple VM images when needed. It supports on-demand resource allocation to adjust dynamically and to ensure the allocation of resources is up to user satisfaction. It exploits parallelism for reducing the execution time and energy consumption of the applications. The VM manager and parallel processing module in cloud manages the smartphone VM and splits and distribute the task automatically to multiple VMs. The future directions of this framework are (1) to improve the efficiency of data transfer for remote code execution by combining static code analysis and data caching (2) to extend the compiler to support automatic properties which reduce the burden of application development and (3) to improve application parallelization.

Kemp et al., [9] proposes Cuckoo framework is named after cuckoo bird, which offloads its egg brooding to other birds. Likewise, the cuckoo framework offloads its computation intensive part to remote commercial or private mini cloud servers. Cuckoo is purely targeted at Android platform, and includes a runtime system, a resource manager application deployed in user smartphone to discover the registered available resources, a programming model which is integrated in eclipse build system and android framework developers to ease and automate large parts of the development process. This model even works on connectivity drops, performs local and remote execution and bundles both local and remote executables in a single .apk file. Instead running complete clone of the smartphone in the remote cloud, it runs a temporary clone having only the service used by the application. As a future direction, the context information of the remote cloud such as processor speed, available memory in addition to the QR code address will be augmented. Additional context information such as mobile devices location, network status is also included. To ensure the secure communication between the smartphone and remote cloud resources, security measure needs to be taken.

H. Flores et al., [10] makes use of the advantageous dynamic variable of cloud computing like performance metrics, task parallelization, and elasticity, the EMCO framework is proposed. It provides a strategy as a solution to overcome the problems such as adaptive partitioning of application, offloading decision-making and cloud-aware dynamic allocation of resources. The EMCO framework consists of (1) a decision engine implementing fuzzy logic by considering both mobile and cloud variables (2) a mobile virtualization infrastructure having virtual instances of mobile devices for automating the process the fetching and storing the code offloading traces into the cloud storage. The code offloading traces has the following information: device details e.g. bandwidth, information about mobile

applications and its components, components execution time and the instance type of the execution of the components (3) an evidence-based control system to analyze the code offloading traces using machine learning strategies such as clustering, Neuro-fuzzy etc., The analysis helps to find offloading pattern so that the fuzzy rules can be Google Cloud Messaging (GCM) service is used for sending the push notification to update the fuzzy rule set of a specific device. A partial prototype of the fuzzy logic is used in the EMCO framework. The future directions of this paper are to fully implement the current mobile cloud simulation, to explore and compare the machine learning strategies for parallel utilization of the rules to enrich the fuzzy logic engine.

Gordon et al., [11] presented the code offloading in COMET framework is concentrated on how to offload the code rather on what and when to offload. COMET runtime system let the unmodified multi-threaded mobile app to run on multiple cloud servers. Based on the workload, the thread migrates between these cloud servers. It is built on the top of Dalvik virtual machine. It implements the distributed shared memory (DSM) by making use of the runtime system's underlying memory model. Virtual machine synchronization primitive is utilized by the COMET system for effective migration. The design aim of the COMET system is to build the framework executing the multithread program with improved computation speed and resistance when network or server failures.

In **Borcea et al., [12]** an avatar prototype has been built and executed on Android devices and Android x86 virtual machines to achieve the goals such as effective execution with fast, scalable, reliable and energy efficient mobile distributed computing on new cloud architecture. An avatar is a software entity, one for each mobile device user reducing workload, storage, and bandwidth. It is instantiated as VMs in the cloud for isolating the resources and to simplify the per-user management of resources. It runs unmodified app components by running the same operating system on both mobile devices and on VMs in the cloud server. Even when mobile devices are offline, the avatars will be available for use.

Heungsik Eorn et al., [13] proposed MALMOS (Machine Learning-based Mobile Offloading Scheduler) framework with online training. MALMOS can be applied to any types of mobile offloading framework. In this work, the MALMOS is applied to DPartner, a Java-based offloading-capable code refactoring framework. Since the mobile applications generated by Dpartner depends on static input for deciding whether to execute the offloading computation locally or remotely. It is combined with MALMOS to take the dynamic offloading decision without any user input. The MALMOS architecture consists of following four modes: computation dispatcher, runtime scheduler, machine learning classifier and the trainer for the machine learning classifier. In the adaptive online training mechanism, responsibility is to dispatch and forward the offloading computation to either remote or local unit. The scheduler job is to request the machine learning classifier for offloading decision (remote or local) by sending the size of the offloadable data and network bandwidth. The machine learning classifier trainer updates the classifier at runtime by feed backing the performance by comparing between the offloaded and local processing. The comparison is evaluated with three algorithms: instance-based learning, perceptron, and naïve Bayes in terms of overhead in training and classification time. Also, the adaptability of MALMOS to various network conditions and computing power on the remote resources is compared with the threshold and linear equation-based scheduling policies. The result of the evaluation shows that MALMOS is 10.9% to 40.5% highly accurate than the two static scheduling policies.

Huijun Wu et al., [14] proposed a model which uses many-to-many mobile cloud service composition which consists of multiple surrogates (cloud service providers), mobile devices and their services like computation offloading and storage. Due to the involvement of mobile devices, there occurs a problem when the mobile device moves from one location to another and also its poor resource constraints. This can be managed and appropriate surrogates are considered by various service metrics for allocating service through mobile cloud service topology reconfiguration. Thus the research focuses on how to manage the services that have been implemented on several service provider. The service composition topology reconfiguration process is modeled theoretically to deal with a series of decision instead one-time decision. Three algorithms are presented to solve the decision process on three mobile cloud application scenarios: (a) finite horizon process for executing certain time period application, (b) infinite horizon process for no clear boundary execution period of application, (c) large state situation for ad hoc, many-to-many service mapping and parallel computing application.

In **Ragib Hasan et al., [15]**(Aura)**,** Building cloud infrastructure having data centers located nearer to the clients are highly expensive and infeasible to operate. So that, the Aura a lightweight system is proposed for computation outsourcing by building the ad-hoc cloud with low power IoT devices. Aura comprises the mobile device with M-Agent (Android Application - WordCount), a controller (a java application) and virtual IoT device (running Contiki OS ported with MapReduce framework. When an IoT device enters and joins the Aura network, it has to multicast the entrance-advertisement (device-live-time, pricing, CPU speed, RAM and Flash Drive) to the controller. The given details are received and passed by the controller whether to accept or discard the message. If the IoT device is accepted, it is acknowledged by the controller and get connected with it., After sometime Mobile device with M-Agent enters building-1 with the job to be outsourced. By joining the Aura network, it makes a job advertisement upon receiving the advertisement, the controller passes it to determine job details (job title, completion time, interrupted computation) and consults with the virtual IoT devices whether to take the job from the M-Agent or not. On acceptance, the controller sends price quotation for executing the task. Based on it the M-Agent agrees on the quotation and submits the job. Then controller splits the task into sub-task and assigns it to various IoT devices based on its capability and pricing. Then the results sent back to the M-Agent.

Pengfei Yuan et al., [16] presented a Uniport framework which is a Uniform Programming support framework for developing Mobile Cloud applications. The framework has MVC pattern architecture to apply the various applications on multiple platforms, a set of programming primitives and

runtime libraries for creating new mobile cloud applications. It also transforms the existing application to mobile cloud applications, a set of development tools such that code generator to generate code skeleton for various mobile platforms and a static analyzer for analyzing the existing applications and to check any constraint violation against Uniport architecture. A case study is performed for three existing mobile application (kigomoku, Fivestones, and GomokuPro) on ios, android, and windows phone to their mobile cloud version only with few lines of code modification (3% to 9%), results show that the transformed mobile cloud applications improve performance times in 3-7X and reduce energy consumption and execution time.

Pelin Angin et al., [17] presented the design of a prototype of a dynamic performance estimation model for offloaded computation in order to provide optimal performance (tracking and relocation) under different cloud hosts. Instead of relying on the cloud platform for all the computation process, it reduces the burden on the cloud by partitioning the applications in the mobile platform itself. The estimation model is integrated into autonomous agents for enhancing the self-performance evaluation. In the computation offloading process, a mobile application is installed; the execution manager in the mobile device communicates the cloud director service for available offloading application modules in the cloud host. Then an execution plan is created with offloading decision by the execution manager. If the result of the decision is offloading, then a bridge is formed between the mobile application module and cloud host. Then the necessary input parameters are migrated, processed and the output data are returned to the mobile device. Experiments are done with two applications (Face recognition and N-Queens). Results show high performance and minimized execution time.

Zohreh sanaei et al., [18] proposed a hybrid mobile cloud computing framework (HMCC) is proposed when the capability of coarse-grained resources (giant clouds with high scalability and low proximity), medium-grained (cloudlets with medium scalability and proximity) and fine-grained (smartphone with high proximity and low scalability) are augmented and interconnected by the wireless and wired network. It focuses on improving the computation offloading energy efficiency and responsiveness. Experiments are done on three prototyped applications which are compared with following compute-intensive tasks namely factorial, X power Y and prime generator. Results show that 80% - 96% response time (RTT) and 83% - 96% energy is saved when using HMCC framework.

Suganya et al., [19] paper focuses on two problems (1) whether the prevailing environment promotes computation offloading to the cloud or not and (2) if offloaded, how it is been partitioned and executed securely. The proposed design framework has a dynamic unsupervised decision maker based on Self Organizing Map for offloading the compute-intensive part of the mobile application in the dynamically changing environment by determining the performance of the application with on-device and on-server. In addition, it also augmented the security mechanism for securely outsourcing the offloaded data using steganography and encryption. During the offloading process, when an application is encountered with method calling instruction, the m/interceptor saves the current method's information, intercepts all the input parameters and serializes and sends it to the encoder. Meanwhile, the fproctor profiles the system environment and resource information. By analyzing all the information, the SOM classifier makes the offloading decision. If data has to be offloaded, it will undergo steganography and encryption process before offloading and send it to the cloud server. There, the cInterface receives and decrypts it. The relevant processing takes place and results will send to the mobile device.

Ra et al., [20] paper focuses on what compute-intensive task to offload and how to perform the parallelism on interactive perception application, to improve the throughput and makespan of such applications. Since interactive e perception applications are not quick in reacting to changes in input complexity, device capability, or network condition, cannot achieve both low makespan and high throughput, has high computation and communication overhead, Odessa is proposed. Odessa is a lightweight runtime which automatically offloads the compute-intensive task with low makespan and parallelized decision for interactive perception application using incremental greedy strategy at runtime. Results show that the performance of the proposed runtime is more than 3x speed with varying execution environments.

Sen Yang et al., [21] paper proposed and implanted mobile application computation-offloading mechanism based on the R-OSGi framework. During offloading, for enhanced performance, it is necessary to minimize the total execution time. Therefore, the paper focuses mainly on how to transfer offloadable modules optimally. Two applications are considered in terms of simple-chain and general. While offloading, an application is represented into directed tree graph with multiple modules. Then, a combinational optimization problem is formulated and using two algorithms, the solution to the problem how to offload the modules is obtained. For performance evaluation, the proposed algorithm was implemented and examined on R-OSGi framework. Results show that, for compute-intensive application with few inputs, the modules are offloaded to the server whereas for the data-intensive application, if the transmission delay of data is small, it is processed at server otherwise it will be processed at the mobile device itself.

Ying Zhang et al., [22] presented a tool called DPartner is proposed in the paper for automatically refactoring an Android mobile application by implementing the on-demand computation offloading design pattern to offload the compute-intensive tasks to the server by importing execution time and battery power consumption. In order to overcome difficulties in offloading the applications such as (1) to identify which parts cannot be offloaded, (2) to determine which parts are worth offloading, (3) adaptation of changes in user requirements and runtime environments (eg. Remote server become unable, unstable network connection etc.,). It has designed computation offloading pattern which refactors or restructures the given code into offloadable one with altered external functionality. The steps of refactoring are: (a) detecting movable classes, (b) classes are able to offload, (c) detecting classes which are offloaded as a whole, and (d) parallelizing deployable files. In other words, the DPartner transforms the analyzes bytecode to discover the movable parts of an application (offloadable), then rewrites it by applying the design pattern

and finally generates two deployable files onto the smartphone (.apk) and the server(jar files). On the evaluation of the three application such as Linpack, Andgoida, and XRace, shows that the reduced execution time with 46-97% and power consumption with 27-83%.

Mahbub E Khoda et al., [23] presented a design for an intelligent code offloading decision-making system called ExTrade. The ExTrade system will ensure improvement in both response time and energy consumption of the mobile application. Before offloading the clone of the mobile application has to be launched in the cloud server and necessary updates should be done simultaneously. The offloadable threads are annotated by the developer. The decision maker using Lagrange multiplier, a nonlinear optimization solver is used to decide whether to offload the code or not by analyzing whether it meets computation and energy criteria. The profiler feeds the environment and system information to the execution handler or decision maker. The statistical regression-based model estimates the execution time of the behavior of the environment and application usage. Client handler in the cloud side is for handling control request (to obtain connectivity information) and task execution request (sending application id and thread id to the cloud server). Two classes of application: Heavy computation and less data transfer (N-Queens) and Heavy computation and heavy data transfer (face detection) are deployed to the device for analyzing the performance. For measuring the power consumption of the applications, power tutor application is used. The experimental result shows that the system is improved performance in processing the computation, accuracy prediction, and energy consumption.

The objective of **H. Flores et al., [24]** is to focusing on overcoming the challenges in practical usage of computation offloading such as inaccurate code profiling, integration complexity, dynamic configuration and scalability in offloading. It offloads the code at method level using java reflection. The code profiler information is based on JSON schema by using the following details: candidate method's name, device latency and server type to be offloaded, code execution plan (parallelize the code into n processes) and additionally user location by gathering cache results based on location. To execute multiple applications concurrently a special compiler is based on the server. Auto-scaling mechanism is implemented so that during multiple offloading requests, the load of the subscriber is split between other available servers. Results on pre-cached case provide low response time and energy consumption.

Bowen Zhou et al., [25] presented a framework adapting client-server communication model. It uses context-aware offloading decision algorithm which selects suitable wireless medium (cellular/WiFi using TOPSIS technique) and cloud resources (MANET cloudlet and public clouds) based on different context available on the mobile devices. The framework consists of three core components: (a) context monitor which has device profiler to collect the hardware information and send it to the cost estimation model, Network monitor to gather mobile device and network context and pass it to cost estimation model, program profiler to track the program execution at method level and stored in mobile database which will be used by cost model for prediction, (b) decision engine which has cost estimation model to determine the execution cost of offloaded task and it uses context-aware decision-making algorithm to make decision of when, where and how to offload the task, (c) communication manager to discover and handle the communication between client mobile device and in either device cloud/ remote cloud VMs. Experimental results in calculator and face detection application show that the prototype reduces the execution cost for heavy computation up to 70%.

Yaser Jararweh et al., [26] aimed to reduce the power consumption and network delay when offloading the mobile application compute-intensive task to the cloud. It utilized the cloudlet based mobile cloud computing system which consists of a mobile device, set of cloudlets and an enterprise remote cloud. The flow of execution is: whenever a task to be offloaded from mobile device contacts the cloudlet which forwards it to the Enterprise Cloud (EC) server. This research has experimented with three mobility scenarios by sending 226 kb file size to both cloudlet and Enterprise cloud. (a) Mobile device access the EC through clodlet1 until the job is completed on cloudlet1, (b) mobile device access EC through cloudlet1 but before job completion moves another location, there it accesses Enterprise Cloud through cloudlet2. The data and remaining process details in cloudlet1 are sent to cloudlet2 on request, (c) here mobile device access EC through cloudlet2 and moves to another location with no cloudlet coverage before job completion. On such scenario, the mobile device makes use of 3G/LTE connection to access the EC. The management of remaining data or processes of incomplete job are sent by two approaches: (1) Centralized approach where EC is responsible for storing the tracking information (current status, connection type, current services, current files, recent cloudlets and incomplete jobs) and (2) De-centralized approach where the mobile device is responsible for managing the tracking data (Recent cloudlets, incomplete job). The result shows that cloudlet outperformed the EC in terms of power, access time and throughput.

Abhirup Khanna et al., [27] proposed offloading model offloads an application based execution pattern of its task. The framework of the proposed research has following components: a device profiler divides the application into numerous tasks and annotates it (CPU/IO), an application analyzer to calculate cost of local and remote execution, a network profiler to determine the transmission cost, a decision engine which receives information from application analyzer, network profiler and performs mathematical calculation to decide whether to offload or not. The proposed decision algorithm is implemented in Cloudsim. Two computing environments are used VM #0 (mobile device), VM #10 (cloud). Two applications (cloudlet #0 and cloudlet #1) are running on VM #0. On execution, the following two cases happen (a) both applications execute on locally and (b) either of the application run on the cloud. Results show that with increased no. of tasks being offloaded, there is a reduction in execution time to certain value and decreased communication cost.

Queen Kaur et al., [28] focuses to overcome the existing schemes drawbacks where the computation offloading having high energy consumption and communication cost. The proposed work is implemented on Cloudsim by creating two VM nodes representing mobile and server. It checks for the requirement, then selects the computation offloading

scheme algorithm, removes the repeated data using Buffer Allocation Method, offloads the computation to the VM which is closer to the current VM and generate the performance analysis graphs in terms of execution time, CPU workload and energy consumption. During offloading, the Buffer Allocation Method inputs the packet sequences process the packets, if it is valid packets it stores in the buffer then transmit the packets from the buffer to the valid path to the cloud, otherwise the packet will be discarded and the transmission failed. By doing so, the offloading compute-intensive task takes reduced execution time and balanced load on the local device.

Chengke Wang et al., [29] provides a detailed analysis of the cause of energy wastes while the phone is not used. It measures the power consumption using power monitor application, the WiFi traffic is monitored by using WinPcap tool, when the smartphone sends/receiver a network package. To collect the user traces, a lightweight tool is used for monitoring the event traces and resource usage information on smartphones by volunteers. The tool has an event-driven collector to record event traces (occurrences, timestamp), a polling collector to read the information on resource usages such as CPU and network. The proposed solution to optimize the standby energy is to optimize the tail energy screening off all the session manually and and saves 1.3% out of 2%, energy to turn off the unused network connection or switch to another network (WiFi to 3G) which saves 14.3% to 20%, delaying some packages to group together for reducing tail time which is optimized as 66% of energy savings, terminating the background applications automatically by prediction or providing some warning messages which 20.3% of energy. The result shows that based on the user traces, the standby time is optimized overall by 87%.

In **Vikas Pandey et al., [30]**, offloading decision is made at runtime by using depth-first search (topological sorting) for dynamic profiling of each component is calculated at runtime using call graph which consists of nodes and edges where nodes represent the executing module and edges represent the interaction between the two modules. The weight of nodes and edges denotes the execution cost and transmission cost respectively. The algorithm partitions the call graph in to sub-graphs (method call sequence) using DFS. Then to find the best beginning and ending offloading point, linear search is applied on the call sequences. It also considers network bandwidth, state transfer time and remote and local execution time of method when partitioning the application. The performance of the proposed algorithm is better than 0-1 ILP approach, since it offloads all the subsequent methods of the offloaded method. The algorithm also reduces the execution time and power consumption of the mobile phones. This dynamic algorithm's time complexity is $\Theta(E+V)$.

The aim of **Shivani Sachdeva et al., [31] is** to minimize the consumption of energy by deploying the software optimistically in mobile clouds using ACO (Ant Colony Optimization) graph partitioning technique. It makes use of the ACO technique to find the shortest path between the user and remote cloud server and also partition the application by initializing and optimistically deploying the call graph G(V,E) and partition the graph by applying Ant Colony Optimization. Then the pheromones are updated on best energy solution obtained and is repeated until a fulfilling solution is found.

Jianwei Niu et al., [32] improves the performance of partitioning with fixed bandwidth assumption and to avoid the dynamic partitioning overheads by finding periodically the partition of the application online whenever a change in bandwidth. Therefore, it combines both static analysis and dynamic profiling and constructs the weighted object relation graph of the application. Then considering bandwidth as a variable it prepares optimization partitioning model based on execution time and energy consumption. To obtain optimal partitioning solution two bandwidth-adaptive algorithms (Branch-and-Bound based Application partitioning and Min-Cut based Greedy Application partitioning) are proposed for partitioning small and large-scale applications respectively. Eventually, based on the partitioning results, the application components are executed in distributed manner. Experimental results show that both algorithms reduce the consumption of energy and time of executing the mobile applications.

Ioana Giurgiu et al., [33] partitions the application by taking dynamic decision on considering the parameters such as network conditions, mobile device's CPU load, and size of end-user input. This approach continuously profiles the structure of the application's resource requirements and constraints of the mobile devices. After profiling such information, it is deployed dynamically by adjusting to the changes in network conditions, mobile device CPU load and end-user input. It also caches the deployment setting for further execution of the application with different inputs. On comparing various applications, the result of this partitioning approach shows that it achieves 75 percent of performance gain and 45 percent of reduction in power consumption.

Salwa Adriana Saab et al., [34] presented an FSP (Free Sequence Protocol) for dynamic execution and using min-cut algorithm; the application was formulated as a flow graph and partitioned. The algorithm is further extended to augment the security measures (such as 128-bit AES encryption) while offloading FSP-based android mobile application. The main goal of the system is to formulate a mathematical model to optimize the energy consumption problem. The system consists of profiler for measuring the hardware and software requirements and sends the information to the cloud server; decision engine is based on the minimum-cut maximum-flow algorithm. Results revealed the engine minimizes the energy consumption with high performance.

Lei Yang et al., [35] focuses on optimizing the partitioning of data stream application such as QR code recognition to achieve high throughput in processing the stream of data. The framework supports dynamic partitioning of application and provides better scalability. The framework makes use of genetic algorithm for optimal computation partitioning to increase the throughput of the application. Based on the available bandwidths, the partitioning of the algorithm will be executed in the cloud server. Results revealed that the partitioned application's performance is two times better than the throughput of the application without partitioning.

Mahir Kaya et al., [36] proposed a framework based on the invention of control mechanism where the task of object creation is delegated at the offloading factory which then

decides whether to create a proxy or object at the runtime. The advantage of the framework is distribution transparency of offloading. The framework consists of offloading factory for creation and management of proxies of the requested classes, a method call is made to the remote server and appropriate objects are created by the offloading factory using its unique ID, a profiling manager monitors each method call and constructs a call graph with information such as method's execution time, call frequencies, input parameters size and return value size, an optimal decision manager which makes best offloading decision based on call graph's edges and vertices weights. It uses min-cut algorithm with FM heuristic for partitioning the graph. By using algorithm, first the offloadable classes are identified and next vertex with best offloading gain is identified which is then moved as another partition and the steps are repeated until better partition is found. It has a deployment manager for deploying and sending the server-side application to the repository server and a discovery module for available cloud service discovery which in turn initiates the offloading process when the application is started. Result shows that offloading optimal combination of components to the cloud server reduces the processing time and consumption of energy in mobile devices.

Luis D Pedrosa et al., [37] proposed a case for automatic partitioning system which uses heuristic based input complexity metrics to estimate the usage of resources with automated learning procedures to a greedy partitioning algorithm which optimizes the speech recognition library's execution, achieves accurate predictions in resource usage by the application components using complexity metrics and saves 21 percent on energy consumption.

Huin Suo et al., [38] focused on reviewing the security and privacy issues in mobile cloud computing. It analyzes the above said issues in the following three aspects: the mobile terminal, mobile network, and mobile cloud. In the mobile terminal, the security issues are: a) Malware software, which is downloaded automatically unknown to the user while installing any applications which leads to access illegally the personal details or automatic pay without user's knowledge, b) software vulnerabilities where installed application software or operating system bugs will breach the data, c) other causes such as lack of security awareness and mis-operation by many end users. The current solution for the mobile terminals are: a) Detecting and preventing the malware eg: CloudAV which is installed on the cloud side to avoid mobile terminal resource constraints, b) installing the system patches periodically and checking the software integrity, c) Regulating the user's behaviour by not clicking the unexplained link, careful in receiving the data transmission from unknown users, not installing unauthorized /third party software and turning off the Bluetooth or WiFi interfaces when not in usage. In the mobile network, the issues are a) information leakage during data transmission or b) malicious attacks like Denial-Of-Service. The solutions of current approaches for the above said issue are: transferring encrypted data over the mobile network and using security protocols to reduce various network attacks. In the mobile cloud, the issues are: a) susceptible cloud platform, where attacks of stealing valuable information can happen by outsiders, another cloud user, or inside staff of the cloud computing operators and b) Data and privacy protection where the ownership and management of the data are separated and it is stored in shared infrastructure in any place of the world. The solution to the above said issue by current approaches are: a) integrating the cloud with current security technology, b) managing the keys and using encrypted data, c) During transmission, authenticating the data and use access control for protection mechanism.

Solanke Vikas et al., [39] reviewed the security problems in mobile cloud computing. The threats are categorized into five types: (1) Physical threats, (2) Application based threats, (3) Network-based mobile security threats, (4) Web-based threats and (5) Other active threats. In physical the issues are: a) Device possession where privacy information will be revealed if employees use company devices for personal activities and if employee's personal devices are used for business function, b) theft or lost devices by unknown person hands who makes use of that devices for dangerous interactions by revealing bank accounts, social network, contact list etc., In application based threats, the issues are malware, spyware, vulnerable and unlicensed application which targets the mobile device for gathering decision history, text messages, user location, browser history, contact list, email, photos and use this information to change the bill without user knowledge, stopping some important services and transferring malicious applications to the device without user's knowledge. In network-based mobile security threats, the issues such as WiFi sniffing, Denial of service, Session hijacking, Insider attack will take place. In web-based threats, phishing scams, Drive-by downloads, and Browser exploitation issues will happen. Eventually, in other attacks, the issues are: Internet protocol vulnerabilities, information recovery vulnerability, and unauthorized access will take place.

Pelin Angin et al., [40] presented a dynamic computation offloading model based on autonomous agent-based application modules. The agent-based model is augmented with the dynamic self-protecting tamper-resistance approach for dynamic detection and reporting of code tampering in computation offloading. Here, the application modules are treated as the mobile agent which is nothing but the chunk of application code package which will be executable on cloud virtual machine when a mobile application is installed; the execution manager checks the cloud directory service for the list of cloud hosts in Amazon EC2 for execution. If founded, the execution plan with offloading decision will be generated by execution manager. During offloading, a bridge is formed between the calling mobile agent mobile and selected cloud host by execution manager for migrating the offloaded module between mobile and cloud host.

Kilinc et al., [41] proposed an application-specific firewall for android application with some additional functions such as VPN technology (Point to Point Tunnelling Protocol) and cloud-to-device messaging (C2DM) framework to determine the malicious application. In the framework, the cloud monitors the application continuously with its reputation (Good, Bad and Unknown) and compares the traffic with the known susceptible malicious servers. If the application is an unknown one, the internet connection is fetched from the VPN service of the VPN server. Then the VPN monitors the traffic of the data to determine whether it is a malicious one. If it is malicious, the traffic will be blocked.

Xu et al., [42] presented a secure web referral service. Here a Secure Search Engine (SSE) ensure the mobile websites against phishing and Man-In-The-Middle attack. Each user is provided with a VM as personal sec-proxy on cloud-based virtual computing. In each VM, SSE uses the web crawler program to check the valid IP address and certificates. The SSE has the following components: URL services, SSL verifier, Phishing filters, SSE crawler, SSE services, DNS services, and storage services. To counter the web-based MITM attack, SSL verifier is used and to counter the phishing attacks, phishing filter is used with optimized processing time. The result shows that the secure engine is secured without any intrusion. It also saves power consumption than the existing web-based anti-phishing solutions.

Popa et al., [43] proposed a secure framework for transmitting the data between the same mobile cloud application's components. It assumes the applications integrity during the mobile installation process and during updating process between mobile and the cloud. This framework includes five managers for providing secure transmission: mobile manager, the mobile and cloud security manager, the optimization manager, the application manager and the policy manager. Here, the mobile manager collect data events on the mobile and send it to the appropriate manager. The mobile and cloud security manger take care of the composition on both sides. The optimization manager sends the information from network and energy sensors to the mobile manager. The application manager verifies the application integrity at setup process (checking whether the application exists in the stores such as Google store, Apple, Amazon etc.,). The policy manager decides which security components are required for a specific security level of certain data. Thus the framework provides a secured scheme with low energy consumption and component-based security by extending the security properties for preferred data using Https.

R. Chow et al., [44] presented an authentication scheme doesn't require any password, username or biometric data for authentication purpose. Instead, it uses TrustCube mechanism for authentication by generating an authentication score based on user's behavior. It is then compared with a generic threshold value to check whether the client is authentic or not. The authentication score varies for the different application. The authentication scheme consists of four modules: client devices, authentication consumer, authentication engine, and data aggregation. The data generated (browsing history, call records, location details, MMS, SMS, Phone contacts etc.,) by the client device are stored in the local cache and after it will be collected by data aggregator. These context information and authentication policies from authentication consumer are then extracted by authentication engine to respond the client with the authentication results.

Grzonkowski et al., [45] proposed an enhanced authentication scheme using Zero Knowledge Proof technique is presented in their papers to protect from phishing attack and to protect the user's password to the visiting websites. It also ensures that the users are not redirected to another web page after login to the corresponding one. This authentication scheme is called SediCi 2.0 and consists of three modules: Client (C), Services (S), and the Authentication services (AS). Here the client in the client application creates an account in AS with his password to generate the public key. After that, the client registers to the service which in turn verifies and records the client login details. Now, both the login details and public key are sent to AS. To get authenticated the client has to visit the service and should gain Auth_ID from the service. Now the Auth_ID and URI are sent to AS for verification whether the Auth_ID corresponds to the URI. Now if the client sends the login to service, it verifies the client using Auth_ID and if verification is successful, the authentication of the client is completed.

Zhao et al., [46] focuses on biometric encryption technology is incorporated in mobile cloud platform for secured authentication since biometric data are difficult to forget, forge, share or lose. Here user first registers his biometric records in the cloud database. Upon login, the identification phases match the given biometric feature with the stored one.

Itani et al., [47] aimed to provide integrity of mobile device the proposed system consists of three entities: Mobile client, Cloud Service Provider (CSP), and trusted third party (TTP). The work of CSP is to manage, operate and allocation of cloud resources. TTP has coprocessor on the remote cloud which distributes the secret key KS to the mobile client and generates Message Authentication Code (MAC).The system has three phases namely: initialization phase, Data updating phase, and Integrity verification phase. In the incremental phase, for every file block in the mobile device, an incremental MACFX is created using KS and sends it to the mobile client. If both MAC's are equal, the integrity is verified successfully.

S.C. Hsueh et al., [48] provided a scheme for both integrity of mobile device data and authentication of the mobile user. In order to provide security when the data is offloaded to the cloud, the following techniques are used: encryption algorithm, hash function, digital signature, random number and secret value. The framework has following four modules: the mobile device (MD), Cloud service provider (CSP), telecommunication module (TM), and certification authority (A). The role of the mobile user is to upload, download, share and synchronize the data between cloud and mobile user. The certificate authority will authenticate the source. The role of telecommunication is to generate cloud password and to store user information action. The responsibility of the cloud is to store the mobile user personal data. Three keys namely, secret key (SK), public key (PK), and session key (SK) are distributed among the mobile devices, telecommunication mode, and certificate authority securely. To access the cloud resources, the mobile has to register with the cloud via certificate authority which redirects it to the telecommunication module. The TM generates a password for mobile users to utilize the cloud resources. Thus using this privilege and techniques the mobile user uploads and downloads the data from the cloud securely. Eventually, this scheme is less energy efficient and scalable.

Debashis De et al., [50] The objective of this paper is to reduce energy and processing time by offering service through heterogeneous wireless networks. The profiling information are mobile battery power, wireless state and requirements of the application. The offloading decision engine makes use of fuzzy logic and hand off decision

algorithm. Results show that improved execution time and reduced energy consumption.

Amani et al., [51] The aim of this paper is to minimize the energy consumption. The profiling information is fetched from the logger module and application profiler. Genetic algorithm is used for making the offloading decision. Results show that optimum energy and resource utilization but cannot large amount of data.

Benkhelifa et al., [52] This paper alleviates the management burden of offloaded code using autonomous agent-based application partitions. Power tutor and static application profiler are used as profiling techniques. The application is partitioned with agent-based partition using call-graph. For offloading decision execution manager is used with cost model. Results show that improved performance in offloading the task but no quality of service on cloud resources.

Pelin Angin et al., [53] Using online supervised learning algorithm as a decision model, this paper autonomously optimizes the execution of service within the MCC framework. The context data are used as profiling information. The application is partitioned using service selection optimization algorithm and service oriented architecture. Results show that the decision module has become more efficient assigning the task to either the mobile device or cloud resources but no secure communication.

Piotr Nawrocki et al., [54] The objective of this paper is to reduce the makespan and energy using context information. The profiling information are obtained from program profiler, device profiler and network monitor. The application is partitioned using java annotations and reflections. The cost estimation model and context aware decision making algorithm are used for making the offloading decision. Fault tolerance using checkpoints are used in the framework. Results show that the cost of execution time and energy are low using the decision with current context.

Bowen Zhoun et al., [55] The objective of this paper is to provide computation offloading as a service to mobile devices. The application tracker, connectivity predictor and execution predictor are used as profilers. Annotations and java reflection are used for partitioning the application. Greedy algorithm is used for decision making. It enables effective computation offloading at low cost.

Cong shi et al.,[56] This paper is based on the inversion of control mechanism which delegates the object creation task to the offloading factory. History based profiler are used. Remote proxy classes, call graph model are used for partitioning the application. Min-cut algorithms with FM heuristics are used for making the decision. Secure socket layer used for secure connection and single sign-in authentication using OAuth mechanism are used. Results show that the distribution transparency of offloading and the program structure not being changed by the developer.

7. CONCLUSION

Different offloading techniques and application partitioning methods show that there is no technique that improves all the parameters i.e. Execution time, Energy consumption and Communication cost. As a future scope new technique should be designed that improves all the parameters. Hence, an Efficient Code profiler, System profiler, Reasoner, Optimal resource allocator and highly securable mechanism for offloading Real-time mobile applications in a dynamically changing environment by estimating the application's on-device and on-server performance to be introduced for better offloading mobile applications to clouds. So those, even the resource-poor mobile devices can execute resource-hungry applications with limited features and resources.

8. FUTURE RESEARCH DIRECTIONS

To overcome the issues in existing research, a mobile application offloading framework with following capability has to be designed and implemented. A lightweight framework with optimal offloading components such as dynamic Code and System profilers, efficient Application partitioning and reasoning methods that should improve all the parameters in terms of execution time, energy consumption and Communication cost. Additionally, an auto Scaling mechanism has to be implemented for allocating the resources and balancing the load on the cloud server. So that the users' Quality of Experience can be enhanced.

An effective Transmission medium selecting mechanism is required to determine suitable cloud resources based on the network connectivity and bandwidth availability. So that, extended battery life and high-performance gain can be achieved even in heterogeneous wireless environs. The highly securable technique to authenticate, authorize and ensure data integrity of the offloaded computation part, since all the offloaded components are accessed by public cloud servers. Since only testbeds are availed for killer MCC applications in Crowd sourcing, Location-based mobile cloud services, Collective sensing, Augmented Reality and Mobile gaming. An efficient implementation is needed in these areas.

REFERENCES

[1] Wu, Huaming, "Analysis of Offloading Decision Making in Mobile Cloud Computing", Freie Universität Berlin, Diss., 2015.

[2] Yating Wang, Ing-Ray Chen, Ding-Chau Wang, "A Survey of Mobile Cloud Computing Applications: Perspectives and Challenges", Journal of Wireless Personal Communications, Springer, pp. 1607-1623, 2015.

[3] Mohammed A. Hassan, Kshitiz Bhattarai, Qi Wei and Songqing Chen, "POMAC: Properly offloading Mobile Applications to Clouds", In Proc. HotCloud'14 USENIX conference on Hot Topics in Cloud Computing, pp. 7-7, 2014.

[4] G. Chun, Sunghwan Ihm, Petros Maniatis, Mayar Nasik and, Ashwin Patti, "Clonecloud: elastic execution between mobile device and cloud", Proc. Sixth Conference on Computer Systems, pp. 301-314, 2011.

[5] Jieyao Liu, Ejaz Ahmed, Muhammad Shiraz, Abdullah Gani, Rajkumar Buyya, Ashan Qureshi, "Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions", Journal of Network and Computer Applications, Science Direct, Pp. 99-117, 2015.

[6] Muhammad Baqer, Mollah, Md. Abdul Kalam Azad and, Athanasios Vasilakos, "security and privacy challenges in mobile cloud computing survey and way ahead", Journal of Network and Applications, Science Direct, pp. 38-54, 2017.

[7] Cuervo Aruna Balasubramanian, Dae-ki cho, Alec Wolman, Stefan Saroiu, Ranveer chandra, and Paramvir Bahl, “Maui: Making Smartphones Last Longer with Code Offload”, Proc. ACM MobiSys 2010, San Francisco, CA, June 15–18, 2010.

[8] S. Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and, Xinwen Zhang, “Thinkair: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading,” Proc. IEEE INFOCOM, Orlando, FL, pp. 25–30, 2012.

[9] R. Kemp, N. Palmer, T. Kielmann, H. Bal, “Cuckoo: a computation offloading framework for smartphones”, Mobile Computing, Applications, and Services, Springer, pp. 59–79, 2010.

[10] Huber Flores and Satish Srirama, “Adaptive Code Offloading for Mobile Cloud Applications: Exploiting Fuzzy Sets and Evidence-based Learning”, Proc. MCS's13 of 4th workshop on Mobile Cloud Computing and services, pp. 9-16, 2013.

[11] Mark S. Gordon, Anoushe Jamshidi, Scott Mahlke, Morley Mao and, Xu Chen, “COMET: code offload by migrating execution transparently”, Proc. 10th USENIX conference on Operating Systems Design and Implementation, pp. 93-106, 2012.

[12] Cristian Borcea, Xiaoning Ding, Narain Gehani, Reza Curtmola, Mohammad A Khan, Hillol Debnath, “Avatar: Mobile Distributed Computing in the Cloud”, in Proc. 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, San Francisco, CA, pp. 151-156, 2015.

[13] Heungsik Eom, Renato Figueiredo Huaqian Cai, Ying Zhang, Gang Huang, “MALMOS: Machine Learning-based Mobile Offloading Scheduler with Online Training”, in Proc. 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, San Francisco, CA, pp. 51-60, 2015.

[14] Huijun Wu, Dijiang Huang, “MoSeC: Mobile-Cloud Service Composition”, in Proc. 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, San Francisco, CA, pp. 177-182, 2015.

[15] Ragib Hasan, Md. Mahmud Hossain, and Rasib Khan, “Aura: An IoT based Cloud Infrastructure for Localized Mobile Computation Outsourcing”, 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, San Francisco, CA, pp. 183-188, 2015.

[16] Pengfei Yuan, Yao Guo, Xiangqun Chen, “Uniport: A Uniform Programming Support Framework for Mobile Cloud Computing”, in Proc. 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, San Francisco, CA, pp. 71-80, 2015.

[17] Pelin Angin, Bharat Bhargava, Zhongjun Jin, “A Self-Cloning Agents Based Model for High Performance Mobile-Cloud Computing”, in Proc. 8th IEEE International Conference on Cloud Computing, New York City, NY, pp. 301-308, 2015.

[18] Zohreh Sanaei, Saeid Abolfazli, Abdullah Gani, Min Chen, “HMCC : A Hybrid Mobile Cloud Computing Framework Exploiting Heterogeneous Resources”, IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, San Francisco, CA, pp. 157-162, 2015.

[19] V. Suganya, Dr. J.Dhillipan, D. B. Shanmugam, “Dynamic Framework Design for Offloading Mobile Applications to Cloud”, IOSR Journal of Mobile Computing & Application (IOSR-JMCA), pp. 15-18, 2015.

[20] M.R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, “Odessa: Enabling interactive perception applications on mobile devices”, In Proc. of Mobisys, pp. 43–56. ACM, 2011.

[21] Sen Yang, Xiangshun Bei, Yongbing Zhang, Yusheng Ji, “Application Offloading based on R-OSGi in Mobile Cloud Computing”, 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, pp. 46-52, 2016.

[22] Ying Zhang, Gang Huang, Xuanzhe Liu, Wei Zhang, Hong Mei, Shunxiang Yang, "Refactoring Android Java Code for On-Demand Computation Offloading", In Proc. ACM International Conference on Object-oriented programming systems languages and applications, pp. 233-248, 2012.

[23] Mahbub E Khoda, Md. Abdur Razzaque, Ahmad Almogren, Mohammad Mehedi Hassan, Atif Alamri, Abdulhameed Alelaiwi, “Efficient Computation Offloading Decision in Mobile Cloud Computing over 5G Network”, Journal of Mobile Networks and Applications, Springer, pp. 777-792, 2016.

[24] Huber flores, Pan Hui, Sasu Tarkoma, Yong Li, Satish Srirama and Rajkumar Buyya, “Mobile Code Offloading: from concept to practice and beyond”, IEEE Communications Magazine, pp. 80 – 88, 2015.

[25] Bowen Zhou, Amir Vahid Dastjerdi, Rodrigo N Calheiros, Satish Narayana Srirama, and Rajkumar Buyya, “A context sensitive offloading scheme for mobile cloud computing service”, IEEE 8th International Conference on Cloud Computing (CLOUD), pp. 869–876, 2015.

[26] Yaser Jararweh, Loai Tawalbeh, Fadi Ababneh, Fahd Dosari, “Resource Efficient Mobile Computing Using Cloudlet Infrastructure”, IEEE International Conference on Mobile Ad-hoc and Sensor Networks, pp. 373-377, 2013.

[27] Abhirup Khanna, Archana Kero, Devendra Kumar, “Mobile Cloud Computing Architecture for Computation Offloading”, 2nd International Conference on Next Generation Computing Technologies, pp. 639-643, 2016.

[28] Queen Kaur Gill and, Kiranbir Kaur, “A computation offloading scheme for performance enhancement of smart mobile devices for mobile cloud computing”, International Conference on Next Generation Intelligent Systems (ICNGIS), 2017.

[29] Chengke Wang, Yao Guo, Yunnan Xu, Peng Shen, Xiangqun Chen, “Standby Energy Analysis and Optimization for Smartphones”, 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, pp. 11-20, 2016.

[30] Vikas Pandey, Shashank Singh, and Shashikala Tapaswi, “Energy and time efficient algorithm for cloud offloading using dynamic profiling”. Wireless Personal Communications, pp. 1687–1701, 2015.

[31] Shivani Sachdeva and Kamaljit Kaur. "Aco based graph partitioning algorithm for optimistic deployment of software in MCC". In Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on, pp. 1–5. IEEE, 2015.

[32] Jianwei Niu, Wenfang Song, and Mohammed Atiquzzaman, “Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications”. Journal of Network and Computer Applications, pp. 334–347, 2014.

[33] Ioana Giurgiu, Oriana Riva, and Gustavo Alonso, “Dynamic software deployment from clouds to mobile devices”, In Middleware 2012, pp. 394–414, Springer, 2012.

[34] Salwa Adriana Saab, Farah Saab, Ayman Kayssi, Ali Chehab, and Imad H Elhajj. “Partial mobile application offloading to the cloud for energy-efficiency with security measures”. Sustainable Computing: Informatics and Systems, pp. 38–46, 2015.

[35] Lei Yang, Jiannong Cao, Yin Yuan, Tao Li, Andy Han, and Alvin Chan, “A Framework for Partitioning and Execution of Data Stream Applications in Mobile Cloud Computing”, IEEE 5th International Conference on Cloud Computing (CLOUD), 2012.

[36] Mahir Kaya, Altan Koc¸yigit, and P Erhan Eren. “An adaptive mobile cloud computing framework using a call graph based model”. Journal of Network and Computer Applications, pp. 12–35, 2016.

[37] Luis D Pedrosa, Nupur Kothari, Ramesh Govindan, Jeff Vaughan, and Todd Millstein. “The case for complexity prediction in automatic partitioning of cloud-enabled mobile applications”. Small, pp. 20-25, 2012.
[38] Hui Suo, Zhuohua Liu, Jiafu Wan, Keliang Zhou, “Security and Privacy in Mobile Cloud Computing”, Wireless Communications and Mobile Computing Conference (IWCMC), pp. 655-659, 2013.
[39] Solanke Vikas S., Kulkarni Gurudatt A., Katgaonkar Pawan, Gupta Shyam, “Mobile Cloud Computing: Security Threats”, IEEE International Conference on Electronics and Communication Systems (lCECS -2014), 2014.
[40] Pelin Angin, Bharat Bhargava, Rohit Ranchal, “Tamper-resistant autonomous agents-based mobile-cloud computing”, IEEE Network Operations and Management Symposium (NOMS), 2016.
[41] Kilinc, T. Booth, and K. Andersson, "WallDroid: Cloud assisted virtualized application specific firewalls for the Android OS", in IEEE 11th International Conference on Trust, Security, and Privacy in Computing and Communications, Liverpool, England, pp. 877–883, 2012.
[42] Xu, L. Li, V. Nagarajan, D. Huang, and W. T. Tsai, “Secure web referral services for mobile cloud computing”, in IEEE Seventh International Symposium on Service Oriented System Engineering, Redwood City, CA, pp. 584–593, 2013.
[43] Popa, M. Cremene, M. Borda, and K. Boudaoud, “A security framework for mobile cloud applications”, in 11th RoEduNet International Conference, Sinaia, Romania, pp. 1–4, 2013.
[44] R. Chow, M. Jakobsson, R. Masuoka, J. Molina, Y. Niu, E. Shi, and Z. Song, “Authentication in the clouds: A framework and its application to mobile users”, in Proceedings of the ACM Workshop on Cloud Computing Security Workshop, New York, pp. 1–6, 2010.
[45] S. Grzonkowski, P. M. Corcoran, and T. Coughlin, “Security analysis of authentication protocols for next-generation mobile and CE cloud services”, in IEEE International Conference on Consumer Electronics-Berlin, Berlin, Germany, pp. 83–87, 2011.
[46] Zhao, H. Jin, D. Zou, G. Chen, and W. Dai, “Feasibility of deploying biometric encryption in mobile cloud computing”, in Eighth China Grid Annual Conference, Changchun, China, pp. 28–33, 2013.
[47] W. Itani, A. Kayssi, and A. Chehab, “Energy-efficient incremental integrity for securing storage in mobile cloud computing”, in International Conference on Energy Aware Computing, Cairo, Egypt, pp. 1–2, 2010.
[48] S. C. Hsueh, J. Y. Lin, and M. Y. Lin, “Secure cloud storage for convenient data archive of smart phones”, in IEEE 15th International Symposium on Consumer Electronics, Singapore, pp. 156–161, 2011., Springer, 2012.
[49] Nawrocki, Bartlomiej Sniezynski ,“Adaptive Service Management in Mobile Cloud Computing by Means of Supervised and Reinforcement Learning”, Journal of Network and systems management, Springer, 2017
[50] Debashis De, Deepsubhra Guha Roy, Anwesha Mukherjee, Rajkumar Buyya, “Application-aware cloudlet selection for computation offloading in multi -cloudlet environment”, Journal of Supercomputing, Springer, 2016
[51] Amani S. Alnezari, Nasser-Eddine Rikli, “Achieving Mobile Cloud Computing through Heterogeneous Wireless Networks”, International Journal of Communications, Network and System Sciences, pp. 107-128, 2017
[52] Benkhelifa, Thomas Welsh, Loai Tawalbeh, Abdallah Khreishah, Yaser Jararweh, and Mahmoud Al-Ayyoub, “GA-Based Resource Augmentation Negotiation for Energy-Optimised Mobile Ad-Hoc Cloud”, 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, pp. 110-116, 2016
[53] Pelin Angin and Bharat Bhargava, “An Agent-based Optimization Framework for Mobile-Cloud Computing”, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, pp. 1-17, 2013
[54] Piotr Nawrocki and Bartlomiej Sniezynski, “Autonomous Context-Based Service Optimization in Mobile Cloud Computing”, Journal of Grid computing, pp. 343-356, 2017
[55] Bowen Zhou, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, Satish Narayana Srirama, and Rajkumar Buyya, “mCloud: A Context-Aware Offloading Framework for Heterogeneous Mobile Cloud”, IEEE Transactions on Services Computing, pp. 797-810, 2016
[56] Cong Shi, Karim Habak, Pranesh Pandurangan, Mostafa Ammar, Mayur Naik, Ellen Zegura “COSMOS: Computation Offloading as a Service for Mobile Devices”, In Proc. ACM international symposium on Mobile ad hoc networking and computing pp. 287-296, 2014
[57] Mahir Kaya, Altan Koc¸yigit, and P Erhan Eren. “An adaptive mobile cloud computing framework using a call graph based model”. Journal of Network and Computer Applications, pp. 12–35, 2016.

