International Journal of Computer Applications (0975 - 8887)
Volume 143 - No.13, June 2016

A Comparative Study of Code Offloading Techniques and Application Partitioning Methods in Mobile Cloud Computing

Amardeep Kaur
Guru Nanak Dev University
Amritsar, Punjab
India (143505)

Kamaljit Kaur
Guru Nanak Dev University
Amritsar, Punjab
India (143505)

ABSTRACT

Mobile cloud computing allows the execution of computation-intensive applications of mobile devices in computational clouds, and this process of executing in cloud by sending the application VM/Components is called application/code/component offloading. Offloading is an effective method to save the execution time and energy consumption of mobile devices. Thus it extends the battery life of mobile devices. Applications are first partitioned into offloadable and non-offloadable components, which are then transferred to remote server for execution. The objective of this paper is to explore the different techniques of offloading and application partitioning methods. These techniques are thoroughly reviewed in this paper. This paper also highlights the comparison of different techniques on the basis of their contribution, merits, demerits and also on the basis of improvement in execution time, energy consumption, communication time.

Keywords

Application Partitioning, Code Offloading, Mobile cloud computing, Energy Consumption, Execution Time

1. INTRODUCTION

Today is the era of smart phones. Smart phones have become necessity in everyone's life. Smart phone's capability of running multiple applications have attracted everyone. But some computation intensive applications consume lots of energy. And energy is always a constraint in mobile devices due to slow development of battery. The execution time of computation intensive applications in mobile devices is also high. So instead of the executing such applications locally in mobile devices, some kind of mechanism is required that helps to reduce execution time and energy consumption.
One such mechanism given by many researchers is mobile cloud computing. Mobile cloud computing is a union of three technologies– mobile internet, mobile computing, cloud computing[10]. Mobile cloud computing is workable by three major components: mobile devices(such as smart phones, PDA, tablets), network technology(such as Wifi, Wimax, 3G, 4G, and LTE) and cloud(such as Microsoft Azure and AWS).
Mobile cloud computing is set of techniques that use near unlimited cloud resources to empower mobile applications .i.e. Computation intensive applications of mobile devices are offloaded to cloud resources for execution [8].
Offloading the applications to cloud reduces the burden of mobile devices of executing applications. Offloading helps to preserve energy consumption in mobile devices and reduce the execution time. Applications are offloaded and executed in cloud resources and result are returned back. Offloading the applications to cloud successfully requires the high network bandwidth and low link failure rate.
Offloading can be done in two ways– either the application/Virtual machine is moved to remote server for execution or application is partitioned into components and components are sent to remote server for execution. In former way, network cost is high and when there is requirement to access mobile device sensors, problem occurs and even energy consumption also exceeds due to communication. But in latter way, this problem is reduced as application is partitioned into offloadable and non-offloadable components. Non-offloadable components are components that are not suitable for offloading, as they provide graphical user interface, have to access mobile device sensors, GPS, and network components.
Application partitioning is a mechanism of splitting up the application into separate components, also maintaining the semantics of application. These partitions can independently operate in distributed environment. The application partitioning is a prephase of computational offloading. Application Partitioning can be done in different levels of granularity:Module level, Method level, Object level, Thread level, Class level, task level, Component level, etc. Application Partitioning is done to improve performance, reusing memory constraints, reducing energy consumption, reducing network overhead, etc.
Energy that code offloading saves depends on the network bandwidth, the amount of data to be transmitted, and amount of computations to be performed [18].
The rest of paper is organized as follows: Section 2 shows Techinques used in code offloading in mobile cloud computing. Section 3 presents Application partitioning methods. Section 4 talks about Discussion about the research on mobile cloud computing till 2016 and Section 5 describes Comparison of various techniques. Section 6 conclude the whole paper.

2. TECHINQUES USED IN CODE OFFLOADING IN MOBILE CLOUD COMPUTING

2.1 Route based Techniques

—**Cloud path selection for offloading based on AHP and TOPSIS. [35]**
(wu et al., 2012)The aim of selecting cloud-path for offloading is to select the best cloud among the available public clouds that satisfies the given criteria. This technique is multiple criteria based that evaluates many criteria such as speed, bandwidth, price, security, and availability. It makes use of analytic hierarchy process(AHP) for determining weights of the criteria for selecting cloud path. And use the TOPSIS algorithm to get final ranking of available clouds.

—**Mapping and Scheduling of TIG for Code Offloading [1]** (balakrishnan et al., 2013)This paper helps to save energy using Dynamic Voltage and Frequency Scaling(DVFS) during mapping and scheduling stages. Mapping is assigning a resource to a task and scheduling is executing tasks at appropriate operating frequency. Two level genetic algorithm is proposed for these two stages. Inner level is for scheduling and outer level is for mapping process. Applications are partitioned to form several task interaction graphs(TIG), which are offloaded to either cloud resources or surrogates. The execution of scheduled TIG must be within global deadlines. This work successfully schedules tasks with 35 percent energy saving in a mobile device.

2.2 Resource based techniques

There are three types of resources to which code offloading can be done. These are–public cloud, cloudlet, and mobile adhoc network. Public cloud such as Amazon, Google, Microsoft azure. Cloudlet acts as middle layer between mobile devices and cloud. Mobile adhoc network is a group of mobile devices, forming a device cloud.

—**Context sensitive offloading scheme [39]** (zhou et al., 2015)In this paper, code offloading decision is made to find suitable cloud resources to which code will be offloaded based on the wireless network available. Three resources are used in this paper–public cloud, cloudlet, and Manet. The best wireless interface is also selected based on six criteria , such as energy cost, link speed, availability, link quality, monetary cost, and conjunction level of channel using AHP and TOPSIS algorithm. And then suitable cloud resource is selected that utilises minimum execution code, which is estimated using cost estimation model. It achieves significant performance improvement.

2.2.1 Cloud based techniques

—(Shiraz et al., 2015) [30] In this paper, author has proposed an offloading framework, that minimises the number of computation-intensive components, migrated to cloud at runtime. Thus it reduces the overhead induced by the migration of components. This framework is examined under three cases. First, application is made to run on mobile devices, second, traditional offloading methods are used for execution and third, proposed offloading technique is used for application execution. Results of these three cases are compared. Results show that the proposed technique is energy efficient and saves energy consumption by 69.9 percent and also reduces data transmission cost by 84 percent.

—(Chen et al., 2015a) [2] Author has optimised the offloading decision algorithm for tasks(independent) of mobile devices using one computing access point and a remote server of cloud. It reduces the overall cost of energy consumption, execution cost and delay. It considers the problem formulation as a non-convex quadratically constrained quadratic program. And to solve this problem, decision algorithm using semi-definite relaxation and a randomization mapping methods is proposed. This algorithm produces optimal performance using computing access point.

—(Pandey et al., 2015) [24] In this paper, author has proposed a dynamic offloading decision algorithm that uses depth first search for calculating the offloading point. Offloading point is calculated at runtime. It uses call-graph. Call graph consists of nodes, which represents methods. Each node has weight that represent execution cost and edge weight represent transmission cost. A call sequence is made from call graph using topological sorting(DFS). After that linear search is performed on all the call sequence to find the best beginning and ending offloading points. This algorithm reduces the energy consumption and execution time. Results show that the performance of the proposed algorithm is better as compared to 0-1 ILP approach. Time complexity of this algorithm is O(E+V).

—(Li et al., 2015) [20] Author has proposed a new approach that performs method level offloading by migrating only required context information for application execution, along with workload. It finds the required memory contexts, before the execution begins by offline decomposing of the executables of application. The results of decomposition are stored along with the executables of application. These results are used by offloading decision algorithm to migrate only required context information to remote cloud server. It helps to save the transmission cost and energy consumption.

—(Chen et al., 2015b) [3] This paper aims to achieve the efficient computation offloading using game theory. Author has proposed decentralised computation offloading game, which is a offloading decision algorithm. In this approach mobile users can self-arrange themselves to obtain offloading decision in decentralised system. This self-arranging characteristics helps to reduce the burden of management of difficult centralised system (assembling of information from multiple mobile users and offloading decision by cloud). In this method, first of all, decentralised computation offloading game is generated, then game is examined in two wireless accesses cases(homogeneous and heterogeneous). Game possesses Nash equlibrium in both the cases. It scales according to the system size.

—(Saab et al., 2015) [27] In this paper, author has proposed a free sequence protocol that execute the application dynamically using call graph. It is further extended to include the security measures. It uses min-cut algorithm for application partitioning. It aims to solve energy optimization problem. It consists of profiler, to measure the software and hardware requirements; decision engine, that provides decision using min-cut maximum flow algorithm; and android mobile app based on FSP. Results show that it saves energy consumption and improves performance.

—(Hung et al., 2012)[15] In this paper, author has proposed mobile application execution framework on clouds. In this technique application on mobile device is first stopped, then the data files containing saved state of application are sent to the cloud and executed in cloud server. Loss of input data is avoided by integrating application replay technique with a state saving scheme. The data synchronisation is prioritised by adding application state saving scheme and categorisation of data. The application saved state is transferred instead of entire VM. It reduces the amount of data to be transferred.

—(Verbelen et al.,2012a) [32] in this paper author has introduced AIOLOS, which is an adaptive offloading decision engine. It is used to consider the resources available in the cloud and varying network conditions dynamically in the proposed decision. It finds location where application partition will be executed by using estimated execution time at both remote cloud and mobile device. This algorithm is computation and energy intensive.

2.2.2 Cloudlet based techniques

—(Fesehaye et al., 2012) [9] In this paper author has presented a design of cloudlet based network and an adaptive decision-based service architecture. It has cloudlet server and mobile devices attached with nearer cloudlet. With the wireless links, mobile devices can exchange information with other mobile devices and server within the network. Using the centralised and distributed approaches two new forwarding algorithms are introduced. Distributed routing algorithm cloudlet carry out routing distributively using local information. In centralised routing algorithm, cloudlet sends its own ID and ID of each attached mobile device to centralised server. Then server generate route by computing and sends it back.

—(Verbelen et al., 2012b) [33] author has presented a fine grained dynamic cloudlet based approach. Any computer system in LAN with accessible required resources is made a cloudlet. Dynamic cloudlet deal with the shortcomings of static cloudlet .i.e. Requirement of deploying a cloudlet infrastructure in LAN by the service provider. This limitation was alleviated by dynamic cloudlet.

—(Gordon et al., 2012) [13] In this paper author has proposed a mechanism to send the multi-threaded applications to nearby servers which is called COMET. It makes use of the machine workload to make decision of thread migration. It is able to migrate any number of threads by using distributed shared storage technique.

2.3 Proxy Based Techniques [16]

(kaya et al., 2016)The proxies of requested classes are created and managed by the offloading factory. Offloading factory manages resource accesses between the cloud and mobile device at runtime. Proxies are created for each offloadable object, and method call is sent to the objects placed in remote server by the offloading factory. Unique Id is associated to the object on server side. If the objects on server side needs the resources of mobile devices then proxies are created that will point to the resources in mobile devices. By creating proxies of all remote resources, offloading factory manages the coordination of access to resources.

3. APPLICATION PARTITIONING METHODS

Application partitioning is pre-offloading process that helps to partition the application. It split up the applications into separate components, maintaining the semantics of application. Each component is able to execute independently in distributed environment. Partitioning is usually done based on graphs. Call graph is generated according to the sequence how the methods call each other. In the same way several graphs are generated in different applications partitioning algorithms according to the level of granularity used such as module level, method level, object level, class level, task level, thread level, component level, etc. [21] . And partitioning can be static(predefined during development) or dynamic (based on runtime conditions).

—**Graph partitioning algorithm based on input complexity metrics [25]** (Pedrosa et al., 2012) Author has proposed a partitioning algorithm based on graph. Application is converted into a graph and then decision is made for partitioning the graph. Complexity metrics are used in this paper to help in predicting the usage of resources by the components of the application. Each component's input (general purpose and application-specific inputs) properties are used to make prediction. These algorithms can easily extend to new application specific of user defined input complexity metrics. It also possesses the property of modularity. This algorithm optimizes the application execution and saves 21 percent on power consumption.

—**Dynamic partitioning approach [11]** (Giurgiu et al., 2012) In this paper, application is partitioned optimally using following parameters: device's CPU load, network conditions, and user inputs. Some components are to be run on cloud and other on mobile devices. Application structure, requirements of resources, and device constraints are profiled continuously. It is a dynamic partitioning approach. After profiling, current application deployment is dynamically reconfigured according to the changes in the network conditions, device CPU load, and user input. It provides 75 percent of performance gain and reduces energy consumption by 45 percent.

—**Bandwidth adaptive partitioning [23]** (Niu et al., 2014) In this paper, static application partitioning approach is enhanced using weighted object relational graphs to avoid the overheads of dynamic partitioning. Application partitioning is optimised by combining static analysis and dynamic profiling in relational graphs. Author has proposed three methods, each having different objectives regarding the optimization of execution time, optimization of energy and their combination. Bandwidth is considered an important parameter that helps in minimising the compromise between energy and time savings, taking in account transmission costs and delay. For small and large scale applications, two application partitioning algorithms are proposed by the author. This approach helps in reducing energy consumption and execution time.

—**Partitioning algorithm for data stream applications [36]** (Yang et al., 2013) In this paper, author has proposed application partitioning algorithm using data flow graph. It makes use of genetic algorithm for increasing the application throughput. This approach provides dynamic partitioning and computation requirements are shared among multiple users in cloud. Cloud resources are efficiently used. It is scalable approach. Partitioning algorithm is executed in cloud. It provides high performance improvement.

—**ACO based graph partitioning algorithm [28]** (sachdeva et al., 2015) In this technique, ant colony optimisation (ACO) is used to find the shortest route between remote cloud and mobile device, and also to partition graph. If first initialises and deploy graph G(V,E) and then apply Ant colony optimisation to partition graph. The updation in pheromones is done using optimal solution and is iterated until a fulfilling solution is obtained and stop condition is applied.

— **Linear time heuristic for partitioning a graph [16]** (kaya et al., 2016)Linear time heuristic for graph partitioning is proposed by FM(fiduccia and mattheyses 1982) so called FM heuristic. In FM heuristic for graph partition, graph min-cut algorithm is given. The best offloading decision is generated based on weights of edges and vertices. FM heuristic only use edge weight for estimating gain. In this algorithm, first offload able classes are identified and then next step is to find vertex with best offloading gain. This vertex is then moved to another partition. This is repeated until no better partition is obtained.

— **Improved (K+1) coarse partition algorithm [26]** (qin et al., 2012)Improved (K+1) coarse partition algorithm is used to partition cost graph. Coarse partition algorithm is improved to include the concept of context awareness. Computing resources, battery power, sensing ability, network connection , all represent context of mobile device. Input and context information is passed to partition module, and application is partitioned and executed But if during the process context changes, then partitioning is performed again to achieve context awareness.

—**Graph partitioning algorithm [34]**
(verbelen et al., 2013)This partitioning algorithm is used for allocating the components of applications to remote server in public cloud. It minimises the required bandwidth between the components of application, but does not minimise execution time of tasks. It generates partition based on infrastructure heterogeneity using multilevel KL based algorithm. It uses the combination of simulated annealing(SA) algorithm and multilevel (MLKL) graph partitioning algorithm to generate hybrid partitioning algorithm. This combination addresses the issue of randomness of simulated annealing. The important aspect of this algorithm is that a better solution is doing with simulated annealing at coarsest level o shortest time and extra computation cost remains low. This hybrid algorithm also makes sure that no cloud server gets overloaded.

4. DISCUSSION

On analysing the research done on mobile cloud computing, it is determined that mobile cloud computing has undergone an evolution, starting from the year 2009 till now. Many researchers have provided improvements, new techniques and new methods that led to this evolution. This evolution can be apparently known from the figure 1. This figure 1 has shown how improvements and new techniques evolved according to the year of publication. The berief introduction of the techniques in figure 1 are summarized as follows:
(Liu et al.,2009)[22] Instead of scheduling tasks, scheduling of ubiquitous mobile service cell [17] is done to the adjustable server in cloud using genetic algorithm. (Chun et al.,2009) [5] It provides the execution of application in smartphones, offloaded to the clonecloud, a group of smart phones. After executing application on clonecloud, result is sent back to the smart phones. (Giurgiu et al., 2009) [12] Application is represented in the form of data flow graph showing interactions between different modules. Two partitioning algorithms are proposed: ALL and K-Step, to partition the graph in offline mode and during device's connection with the server. (Zhang et al., 2009) [38] Elastic application comprises one or more weblets, that operate independent of each other. The computation-intensive weblets are migrated from mobile devices to cloud server for execution and other weblets that require the resources of mobile devices are executed on mobile device. (Satyanarayan et al.,2009) [29] Author introduced the concept of cloudlets. In this framework, mobile device send its workload to a nearby cloudlet, which consists of various resource rich computers, that are connected to remote cloud server. (La et al.,2010) [19] Author proposed a newframework that provides context aware mobile services. (Chun et al., 2010) [6] It provides application partitioning between device and cloud dynamically and it faces the issue of heterogeneity. Dynamic application partitioning also deals with the shortcomings of static application partitioning. (Cuervo et al.,2010) [7] It provides a framework, MAUI that considers energy efficiency while offloading code to the remote server for execution. It ia a fine grained offloading approach that reduces the burden on programmer of updating the application. (Studedi et al., 2010) [31] Author proposed Wherestore, a position based data storage for mobile devices connected with the cloud. Position history of mobile devices is used to find out the data that is to be replicated locally. (Huang et al., 2010) [14] It provides a new framework, Mobicloud, that amplifies the behaviour of mobile adhoc network, MANET by considering mobile devices as service points. Communication is augmented by using trust management, risk management, and secure routing in the network. (Kumar et al., 2010) [18] Offloading code from mobile devices to public cloud server reduces the energy consumption of mobile devices. (Zhang et al., 2011) [37] An elastic application is partitioned into one or more component, called weblets, that can be either platform-dependent or platform-independent. It provides dynamic execution of these weblets either on cloud or on mobile device locally, according to the status of device. (Chun et al., 2011) [4] Author designed a system, called clonecloud, that modifies the application of mobile device automatically , for execution in cloud environment. Application is automatically partitioned using fusion of static analysis and dynamic profiling.
The research after 2011,(i.e. 2012 onwards till 2016) is already discussed in the sections of "Techinques used in code offloading in mobile cloud computing" and "Application Partitioning methods".

5. COMPARISON OF VARIOUS TECHNIQUES

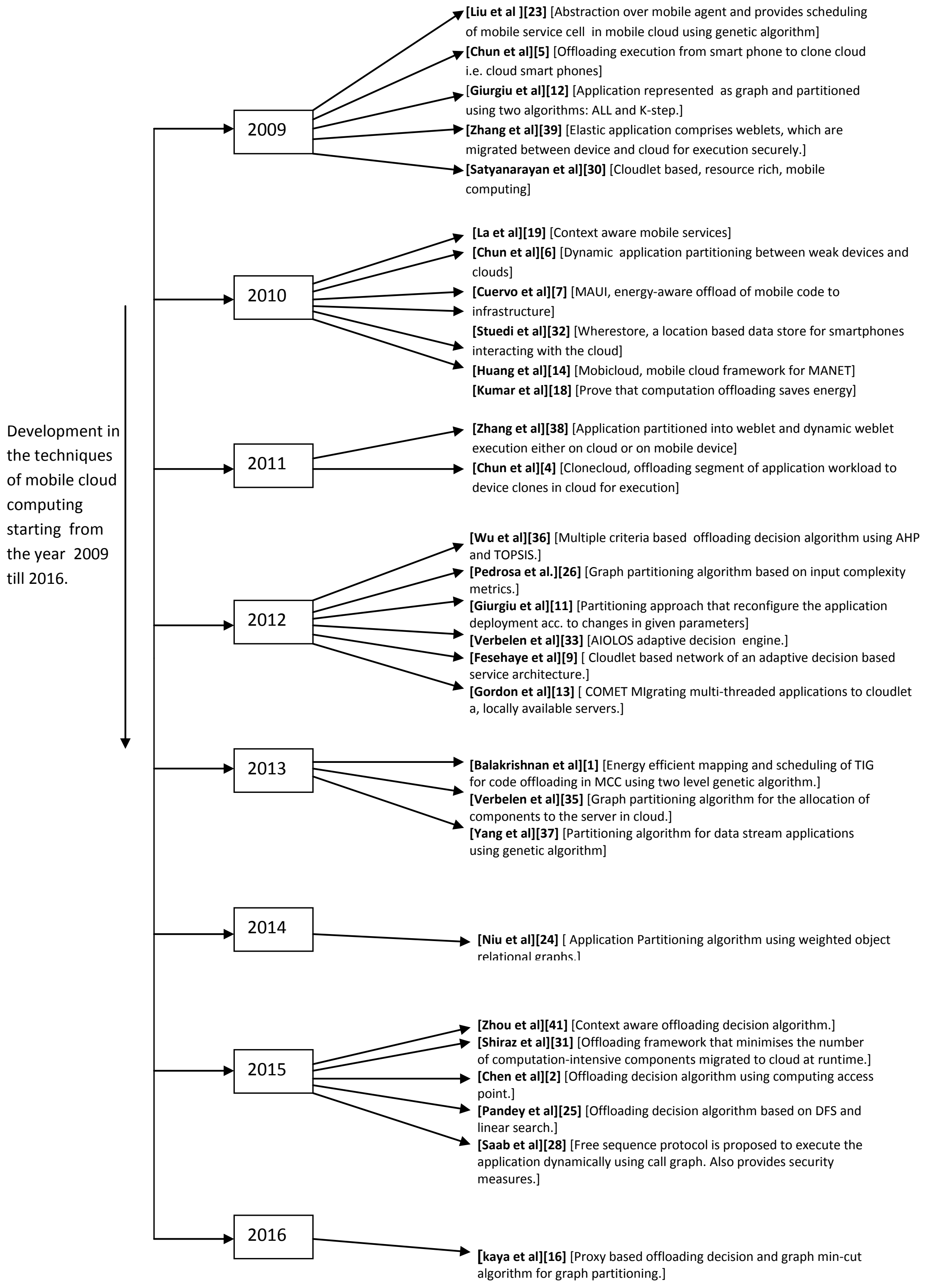
The techniques of offloading and application partitioning are compared in table 1, table 2, table 3, and table 4. In table 1, all offloading techniques are compared based on their contribution, merits and demerits. In table 2, all application partitioning methods are compared based on their contribution, merits and demerits. In table 3, all offloading techniques are compared based on the improvement in execution time, energy consumption, communication time and whether the decision algorithm is simple or complex. In table 4, all application partitioning methods are compared based on the improvement in execution time, energy consumption, communication time and whether the decision algorithm is simple or complex.

6. CONCLUSION

In this paper, a survey is provided on various techniques of code offloading and application partitioning methods. The code offloading techniques help in saving execution time and energy consumption in mobile devices. These techniques are compared on the basis of their contribution, merits, demerits and also on the basis of improvement in execution time, energy consumption, communication time. Different techniques shows different improvements based on the above parameters. The Survey of 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, considering different wireless networks(Wifi, 3G, 4G,.etc.).

7. REFERENCES

[1] Pranav Balakrishnan and Chen-Khong Tham. Energy-efficient mapping and scheduling of task interaction graphs for code offloading in mobile cloud computing.


Development in the techniques of mobile cloud computing starting from the year 2009 till 2016.
2009
2010
2011
2012
2013
2014
2015
2016

[Liu et al][23] [Abstraction over mobile agent and provides scheduling of mobile service cell in mobile cloud using genetic algorithm]
[Chun et al][5] [Offloading execution from smart phone to clone cloud i.e. cloud smart phones]
[**Giurgiu et al][12]** [Application represented as graph and partitioned using two algorithms: ALL and K-step.]
[Zhang et al][39] [Elastic application comprises weblets, which are migrated between device and cloud for execution securely.]
[Satyanarayan et al][30] [Cloudlet based, resource rich, mobile computing]

[La et al][19] [Context aware mobile services]
[Chun et al][6] [Dynamic application partitioning between weak devices and clouds]
[Cuervo et al][7] [MAUI, energy-aware offload of mobile code to infrastructure]
[Stuedi et al][32] [Wherestore, a location based data store for smartphones interacting with the cloud]
[Huang et al][14] [Mobicloud, mobile cloud framework for MANET]
[Kumar et al][18] [Prove that computation offloading saves energy]

[Zhang et al][38] [Application partitioned into weblet and dynamic weblet execution either on cloud or on mobile device]
[Chun et al][4] [Clonecloud, offloading segment of application workload to device clones in cloud for execution]

[Wu et al][36] [Multiple criteria based offloading decision algorithm using AHP and TOPSIS.]
[Pedrosa et al.][26] [Graph partitioning algorithm based on input complexity metrics.]
[Giurgiu et al][11] [Partitioning approach that reconfigure the application deployment acc. to changes in given parameters]
[Verbelen et al][33] [AIOLOS adaptive decision engine.]
[Fesehaye et al][9] [Cloudlet based network of an adaptive decision based service architecture.]
[Gordon et al][13] [COMET MIgrating multi-threaded applications to cloudlet a, locally available servers.]

[Balakrishnan et al][1] [Energy efficient mapping and scheduling of TIG for code offloading in MCC using two level genetic algorithm.]
[Verbelen et al][35] [Graph partitioning algorithm for the allocation of components to the server in cloud.]
[Yang et al][37] [Partitioning algorithm for data stream applications using genetic algorithm]

[Niu et al][24] [Application Partitioning algorithm using weighted object relational graphs.]

[Zhou et al][41] [Context aware offloading decision algorithm.]
[Shiraz et al][31] [Offloading framework that minimises the number of computation-intensive components migrated to cloud at runtime.]
[Chen et al][2] [Offloading decision algorithm using computing access point.]
[Pandey et al][25] [Offloading decision algorithm based on DFS and linear search.]
[Saab et al][28] [Free sequence protocol is proposed to execute the application dynamically using call graph. Also provides security measures.]

[kaya et al][16] [Proxy based offloading decision and graph min-cut algorithm for graph partitioning.]

Fig. 1. Development in the technology of mobile cloud computing starting from the year 2009 till 2016.

Table 1. COMPARISON OF VARIOUS TECHNIQUES OF OFFLOADING ON THE BASIS OF THEIR MERITS AND DEMERITS

s no	Papers	year	Contribution	Merits	Demerits
1	Wu et al. [35]	2012	Multiple criteria based offloading decision algorithm using AHP and TOPSIS.	it evaluates multiple criteria, such as speed, bandwidth, security,price and availability in selection process.	it does not reduce energy consumption.
2	Balakrishnan et al. [1]	2013	Energy efficient mapping and scheduling of TIG for code offloading in MCC using two level genetic algorithm.	It saves 5 percent of energy saving in mobile devices.	It does not reduce the execution time and also faces high communication cost.
3	Zhou et al. [39]	2015	Context aware offloading decision algorithm.	It reduces the execution time and energy consumption.	No mechanism for device failure tolerance.
4	Hung et al. [15]	2012	A cloud based mobile application execution framework.	It reduces the amount of data transfer by only transferring the application saved states and not the entire VM.	It faces delay due to runtime agent program installation, VM creation and states migration.
5	Verbelen et al. [32]	2012a	AIOLOS adaptive decision engine.	It is a dynamic approach that considers the available resources of server and varying network conditions in offloading decision.	This adaptive offloading algorithm is compute- and energy- intensive.
6	Fesehaye et al. [9]	2012	Cloudlet based network of an adaptive decision based service architecture.	The performance of cloudlet based approach is better than cloud based approach.	It suffers from a the initial delays because of service delivery, mobile user authentication and process of joining.
7	Verbelen et al. [33]	2012b	Fine grained dynamic cloudlet approach.	It is a dynamic cloudlet approach which eliminates the limitations of static cloudlet approach. Each cloudlet device share resources dynamically.	It does not provide optimal task mapping.
8	Gordon et al. [13]	2012	COMET MIgrating multi-threaded applications to cloudlet a, locally available servers.	It provides the data consistency across end systems using distributed shared memory and can migrates any number of threads.	It imposes synchronisation overhead across end points.
9	kaya et al. [16]	2016	Proxy based offloading decision and graph min-cut algorithm for graph partitioning.	Distribution transparency of offloading and developer need not change the program structure. It reduces execution time and energy consumption.	It has no mechanism for fault tolerance. This offloading becomes counterproductive if sending classes to remote server have higher network cost than executing them on mobile.
10	Shiraz et al. [30]	2015	Offloading framework that minimises the number of computation-intensive components migrated to cloud at runtime.	It reduces transmission cost by 84 percent and energy consumption cost by 69.9 percent.	It does not consider the problem of consistency when applications are executed simultaneously between mobile devices and remote server.
11	Chen et al. [2]	2015a	Offloading decision algorithm using computing access point.	It helps to reduce the cost of energy consumption, execution cost and delay.	It does not consider the communication cost.
12	Pandey et al. [24]	2015	Offloading decision algorithm based on DFS and linear search.	It is better approach than 0-1 ILP. It reduces execution time and energy consumption.	Communication cost is not considered.
13	Li et al. [20]	2015	Method level offloading technique that sends only required context information for application along with workload.	It saves energy consumption and transmission cost due to least context migration. It keeps offloading effective.	It imposes overhead due to extra time spent on finding the context that is to be migrated.
14	Saab et al. [27]	2015	Free sequence protocol is proposed to execute the application dynamically using call graph. Also provides security measures.	It saves energy consumption and improves performance.	It does not consider the execution time and communication cost of application.
15	Chen et al. [3]	2015b	Decentralised computation offloading decision algorithm using game theory.	This mechanism is efficient and scales as the system size increases.	It does not handle problem of user mobility.

Table 2. Comparison of application partitioning methods on the basis of their merits and demerits

s no	Papers	year	Contribution	Merits	Demerits
1	Sachdeva et al. [28]	2015	ACO based application partitioning algorithm.	It finds the best deployment solution for MCC. It improves the QoS parameters of MCC.	It does not consider executive time and energy consumption.
2	Qin et al. [26]	2012	Improved (k+1) coarse partition algorithm with context awareness.	It provides good performance in different input and network conditions.	No mechanism for fault tolerance, data security and data synchronisation.
3	Verbelen et al. [34]	2013	Graph partitioning algorithm for the allocation of components to the server in cloud.	It minimises the required bandwidth and is dynamic approach.	It does not consider the execution time metric.
4	Pedrosa et al. [25]	2012	Graph partitioning algorithm based on input complexity metrics.	It optimises execution time and saves 21 percent of energy consumption.	It needs programmer's efforts to modify the application.
5	Giurgiu et al. [11]	2012	Partitioning approach that reconfigure the application deployment according to the changes in given parameters.	It provides 75 percent performance gain and 45 percent savings in energy consumption. It is resource efficient.	Applications in this approach needs to be modular. This approach require accurate results of application execution and data transfer before partitioning.
6	Niu et al. [23]	2014	Application Partitioning algorithm using weighted object relational graphs.	It provides reduction in energy consumption and execution time.	This approach is applicable for only specific situations.
7	Yang et al. [36]	2013	Partitioning algorithm for data stream applications using genetic algorithm	It provides improvement in performance and high throughput. This approach is scalable.	It is not applicable for batch computation systems.

Table 3. Comparison of various techniques of offloading on the basis of the improvement in execution time , energy consumption and communiation cost.

s no	Papers	Year	Execution Time	Energy Consumption	Communication Cost	Offloading decision
1	Wu et al.[35]	2012	✓		✓	Simple
2	Balakrishnan et al.[1]	2013		✓		Complex
3	Zhou et al. [39]	2015	✓	✓		Simple
4	Hung et al.[15]	2012			✓	Simple
5	Verbelen et al.[32]	2012a				Complex
6	Fesehaye et al [9]	2012				Simple
7	Verbelen et al. [33]	2012b		✓		Simple
8	Gordon et al. [13]	2012				Complex
9	Shiraz et al [30]	2015		✓	✓	Complex
10	Chen et al. [2]	2015a	✓	✓		Complex
11	Pandey et al.[24]	2015	✓	✓		Simple
12	Li et al. [20]	2015		✓	✓	Complex
13	Saab et al. [27]	2015		✓		Simple
14	Chen et al. [3]	2015b	✓	✓		Complex

Table 4. Comparison of Application partitioning methods on the basis of the improvement in execution time , energy consumption and communiation cost

s no	Papers	Year	Execution Time	Energy Consumption	Communication Cost	Offloading decision
1	kaya et al. [16]	2016	✓	✓		Complex
2	Sachdeva et al. [28]	2015				Simple
3	Qin et al.[26]	2012	✓			Simple
4	Verbelen et al. [34]	2013		✓	✓	Complex
5	Pedrosa et al. [25]	2012	✓	✓		Complex
6	Giurgiu et al. [11]	2012		✓		Complex
7	Niu et al. [23]	2014	✓	✓		Complex
8	Yang et al. [36]	2013	✓			Simple

In *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 34–41. IEEE Computer Society, 2013.

[2] Meng-Hsi Chen, Ben Liang, and Min Dong. A semidefinite relaxation approach to mobile cloud offloading with computing access point. In *Signal Processing Advances in Wireless Communications (SPAWC), 2015 IEEE 16th International Workshop on*, pages 186–190. IEEE, 2015a.

[3] Xu Chen. Decentralized computation offloading game for mobile cloud computing. *Parallel and Distributed Systems, IEEE Transactions on*, 26(4):974–983, 2015b.

[4] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*, pages 301–314. ACM, 2011.

[5] Byung-Gon Chun and Petros Maniatis. Augmented smartphone applications through clone cloud execution. In *HotOS*, volume 9, pages 8–11, 2009.

[6] Byung-Gon Chun and Petros Maniatis. Dynamically partitioning applications between weak devices and clouds. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, page 7. ACM, 2010.

[7] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62. ACM, 2010.

[8] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84–106, 2013.

[9] Debessay Fesehaye, Yunlong Gao, Klara Nahrstedt, and Guijun Wang. Impact of cloudlets on interactive mobile cloud applications. In *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International*, pages 123–132. IEEE, 2012.

[10] Keke Gai, Meikang Qiu, Hui Zhao, Lixin Tao, and Ziliang Zong. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. *Journal of Network and Computer Applications*, 59:46–54, 2016.

[11] Ioana Giurgiu, Oriana Riva, and Gustavo Alonso. Dynamic software deployment from clouds to mobile devices. In *Middleware 2012*, pages 394–414. Springer, 2012.

[12] Ioana Giurgiu, Oriana Riva, Dejan Juric, Ivan Krivulev, and Gustavo Alonso. Calling the cloud: enabling mobile phones as interfaces to cloud applications. In *Middleware 2009*, pages 83–102. Springer, 2009.

[13] Mark S Gordon, D Anoushe Jamshidi, Scott Mahlke, Z Morley Mao, and Xu Chen. Comet: Code offload by migrating execution transparently. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 93–106, 2012.

[14] Dijiang Huang, Xinwen Zhang, Myong Kang, and Jim Luo. Mobicloud: building secure cloud framework for

mobile computing and communication. In *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*, pages 27–34. Ieee, 2010.

[15] Shih-Hao Hung, Chi-Sheng Shih, Jeng-Peng Shieh, Chen-Pang Lee, and Yi-Hsiang Huang. Executing mobile applications on the cloud: Framework and issues. *Computers & Mathematics with Applications*, 63(2):573–587, 2012.

[16] Mahir Kaya, Altan Koçyiğit, and P Erhan Eren. An adaptive mobile cloud computing framework using a call graph based model. *Journal of Network and Computer Applications*, 65:12–35, 2016.

[17] Phyoung Jung Kim and Young Ju Noh. Mobile agent system architecture for supporting mobile market application service in mobile computing environment. In *Geometric Modeling and Graphics, 2003. Proceedings. 2003 International Conference on*, pages 149–153. IEEE, 2003.

[18] Karthik Kumar and Yung-Hsiang Lu. Cloud computing for mobile users: Can offloading computation save energy? *Computer*, (4):51–56, 2010.

[19] Hyun Jung La and Soo Dong Kim. A conceptual framework for provisioning context-aware mobile cloud services. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 466–473. IEEE, 2010.

[20] Yong Li and Wei Gao. Code offload with least context migration in the mobile cloud. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 1876–1884. IEEE, 2015.

[21] Jieyao Liu, Ejaz Ahmed, Muhammad Shiraz, Abdullah Gani, Rajkumar Buyya, and Ahsan Qureshi. Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions. *Journal of Network and Computer Applications*, 48:99–117, 2015.

[22] Qingfeng Liu, Xie Jian, Jicheng Hu, Hongchen Zhao, and Shanshan Zhang. An optimized solution for mobile environment using mobile cloud computing. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on*, pages 1–5. IEEE, 2009.

[23] Jianwei Niu, Wenfang Song, and Mohammed Atiquzzaman. Bandwidth-adaptive partitioning for distributed execution optimization of mobile applications. *Journal of Network and Computer Applications*, 37:334–347, 2014.

[24] Vikas Pandey, Shashank Singh, and Shashikala Tapaswi. Energy and time efficient algorithm for cloud offloading using dynamic profiling. *Wireless Personal Communications*, 80(4):1687–1701, 2015.

[25] 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*, 20:25, 2012.

[26] Zhuoran Qin, Jixian Zhang, and Xuejie Zhang. An effective partition approach for elastic application development on mobile cloud computing. In *Advances in Grid and Pervasive Computing*, pages 46–53. Springer, 2012.

[27] 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*, 8:38–46, 2015.

[28] 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*, pages 1–5. IEEE, 2015.

[29] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *Pervasive Computing, IEEE*, 8(4):14–23, 2009.

[30] Muhammad Shiraz, Abdullah Gani, Azra Shamim, Suleman Khan, and Raja Wasim Ahmad. Energy efficient computational offloading framework for mobile cloud computing. *Journal of Grid Computing*, 13(1):1–18, 2015.

[31] Patrick Stuedi, Iqbal Mohomed, and Doug Terry. Wherestore: Location-based data storage for mobile devices interacting with the cloud. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, page 1. ACM, 2010.

[32] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Aiolos: Middleware for improving mobile application performance through cyber foraging. *Journal of Systems and Software*, 85(11):2629–2639, 2012a.

[33] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Cloudlets: bringing the cloud to the mobile user. In *Proceedings of the third ACM workshop on Mobile cloud computing and services*, pages 29–36. ACM, 2012b.

[34] Tim Verbelen, Tim Stevens, Filip De Turck, and Bart Dhoedt. Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. *Future Generation Computer Systems*, 29(2):451–459, 2013.

[35] Huaming Wu, Qiushi Wang, and Katinka Wolter. Methods of cloud-path selection for offloading in mobile cloud computing systems. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 443–448. IEEE, 2012.

[36] 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. *ACM SIGMETRICS Performance Evaluation Review*, 40(4):23–32, 2013.

[37] Xinwen Zhang, Anugeetha Kunjithapatham, Sangoh Jeong, and Simon Gibbs. Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. *Mobile Networks and Applications*, 16(3):270–284, 2011.

[38] Xinwen Zhang, Joshua Schiffman, Simon Gibbs, Anugeetha Kunjithapatham, and Sangoh Jeong. Securing elastic applications on mobile devices for cloud computing. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 127–134. ACM, 2009.

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