A Review on Energy Efficient Computation Offloading Frameworks for Mobile Cloud Computing

Queen Kaur Gill  
Student, M.Tech  
Computer Engineering and Technology Department,  
Guru Nanak Dev University, Amritsar

Kiranbir Kaur  
Assistant Professor  
Computer Engineering and Technology Department,  
Guru Nanak Dev University, Amritsar

ABSTRACT

Mobile Cloud Computing is an evolving technology that integrates the concept of cloud computing into the mobile environment. Smartphones are boon in the world of technology but they have certain limitations (e.g., battery life, network bandwidth, storage, energy) when running complex applications which require large computations. Using Cloud Computing in mobile phones, these limitations can be addressed. Certain frameworks have been proposed over the years that can address the issues in mobile cloud computing. These frameworks allow the computation to be offloaded in the cloud environment resulting improvement in the battery life, bandwidth, memory capacity and energy consumption in the smartphones. This paper provides the extensive survey on the various energy efficient frameworks available to offload the computation from the mobile to the cloud environment.

Keywords— Mobile Cloud Computing, computation offloading, static offloading, dynamic offloading, energy consumption

INTRODUCTION

Smart Mobile Devices (SMDs) such as smart-phones and tablets Personal Computers (PCs) have become an important part of daily life. Nowadays, mobile devices are not only used for voice calls but are efficiently able to run heavy and complicated mobile applications using internet. The volume of data being processed by smartphones and complexity of mobile applications are increasing day by day. However, these SMD has certain limitations such as network bandwidth, battery life and storage capacity and processor performance.

With the advancement in technology over the years, SMD provide multi-core processors, large memory, bigger and sharper screens, multiple sensors as well as enormous applications. All these together put heavy burden on the battery life of smart phones.

Cloud computing (CC) provides a wide variety of computing resources from servers and storage to enterprise applications. Cloud computing is a hosting environment that is immediate, flexible, scalable, secure and available. The computing resources from cloud can be easily and quickly accessed and released after use with very less management effort. The concept of CC can be used in mobile applications running on SMDs to boost up their performance. With the integration and support of CC into the complex mobile applications, the term Mobile Cloud Computing (MCC) arises.

MCC provides an infrastructure where the data processing and the data storage of mobile cloud applications occur away from the mobile device and into the cloud and bringing applications and mobile computing to a much broader range of mobile subscribers [1]. One of the key features of MCC is migrating the computation intensive tasks to cloud or servers for their execution and then receiving the results from these servers which is known as Computation Offloading. Mobile Devices take advantage of resource-rich infrastructures by offloading the computation to cloud for saving battery’s energy consumption and hence increasing battery lifetime of mobiles.

Computation offloading allows SMDs to become more capable. In contrast to traditional client-server architecture, where clients always offloads the computation to the server and are completely dependent on it, the computational offloading migrates programs to servers which are outside of the user’s computing environment. The term “surrogate computing” or “cyber foraging” is also used for computation offloading.

Before offloading the computation, various parameters such as battery life, current memory capacity of SMD, network bandwidth and latency, computation execution time, run-time migration cost, data traffic are
considered. By analyzing all these parameters, if the computation offloading results in lower power consumption, then only offloading is allowed, otherwise computation can run on SMD itself.

Over the years, many techniques and frameworks have been proposed in order to make computation offloading feasible. All these techniques are used to alleviate the performance of SMDs and making computation offloading more realistic by providing more storage capacity, bandwidth, more processing power and hence increasing the battery life by reducing energy consumption. This paper surveys all the energy efficient frameworks used for computation offloading from mobile to cloud environment.

**SCOPE OF COMPARATIVE REVIEW**

The main purpose of providing the comparative review is to familiarize readers with the previous and current study on computation offloading techniques for SMDs. The computation offloading allows migration of large computations and complex processing from resource limited devices such as mobile phones to resourceful machines such as servers in clouds. This reduces the application execution time and hence reduces the large amount of power consumption.

This paper provides comparative review of various frameworks or techniques which guide the computation offloading mechanism for Mobile-Cloud applications. The main objective of all the frameworks is to save energy by addressing various limitations of SMDs. The frameworks are characterized based on:

- Computation offloading decision whether static or dynamic
- Core component enabling computation offloading
- Parameters considered during offloading
- Applications that support the framework.

**COMPARATIVE REVIEW**

1) **Computation Offloading Decisions**

The decision for offloading is taken either statically or dynamically. In this paper, various frameworks used for computation offloading are mainly characterized based on decisions taken either statically or dynamically. Literature review of various frameworks is provided based on when the computation offloading decision is taken.

1.1 **Static offloading**

Offloading decision is static, when the program is partitioned during its development. The static offloading approach use the performance prediction models or offline profiling in order to estimate the performance of the system. The application is then partitioned into client and server tasks which may subsequently be executed. This approach is used in [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 and 13]. The advantage of static offloading is low overhead during execution. This approach is only valid if the parameters on the basis of which offloading is to be done are already known.

A static approach is suggested in [2] that divides the program into server and client tasks. The information about the computation time and data to be shared is profiled and cost graph is constructed based on the information and then partitioning into the subtasks is done in order to offload to the server or cloud. The application produces the cost graphs demonstrating energy usage and data transmission. A branch and bound algorithm is used in order to minimize the sum of these parameters.

A framework AIDE [3] is suggested which offloads the application’s intensive computation by searching the appropriate surrogate. The AIDE offers a transparent distributed application deployment framework. The execution profile is the core component which profile two main parameters which are prediction of the resources required by the application in near future and execution history. The sets of min-cut partitions of Java applications are obtained statically. The partitions are evaluated by inserting one node in first partition and remaining nodes in second partition. The codes are then moved from one partition to another iteratively based on their highest connectivity. The communication cost is reduced by connectivity of nodes. These partitions are then migrated and executed on appropriate surrogate. This framework is used to run JavaNote, Biomer etc.
A framework DiETis proposed in [4] which can modify the Java by tecodetoorder to offload themethods. The offloading mechanism used in this framework is automated and portable and hence no special JVM reliant instruction is required. The application is partitioned into client and server parts. The client part is downloaded to SMD when SMD user requests to execute the application. The computation intensive methods are modified statically on SMD using remote procedure calls and send to the server. The server first reads the request and then executes the code. The framework when used in SciMark benchmark, the execution speed increases to 59%.

An application partitioning algorithm [5] is suggested that splits the application into two major parts. The first part called as un-offloaded part constitutes that part of the application that can run locally on the SMD. The second part constitutes k parts or partitions that can be offloaded to the server node. The application components are modeled to form multi-cost graph dynamically demonstrating computation and communication costs. Weights of edges and vertex are used by the algorithm in order to do effective partitioning.

An adaptive technique is presented in [6] that implement the process of computation offloading by utilizing a primary profile attained by program execution; hence the execution profile is maintained. The time-limit for executing the program with in the SMD is decided. If the execution of the program is not completed within a specified time limit, then the program is offloaded to the server. Hence, the offloading takes place after specified time-out. The computation time required to run the application locally on the SMD is obtained and compared with the time required to run the same application on the server node, and significant improvement in execution time is the result. Energy consumption is also reduced when this technique is used in image processing applications.

Wishbone application partitioning algorithm is suggested in [7] which also does profiling. An application is modeled as a data flow graph involving various operators used in the code, in order to run on distributed and dissimilar devices in network having sensors. A sum of network bandwidth and load of CPU is minimized statically during compilation. The Integer Linear programming problem is solved using this technique. WaveScript is used to demonstrate the various inter connections within the components of the application. The programming approach used in Wishbone is not robust and inefficient to handle variations in the sensor networks.

The CloneCloud framework is suggested in [8] which is similar to MAUI [20] framework and overcomes some drawbacks of MAUI by doing the partitioning and reintegration at the application level statically. The partitioning phase of the framework includes static analysis of different execution conditions on both SMD and the cloud node, dynamic application profiling and optimization solution. The CloneCloud uses a process-based approach by generalizing the pieces of binary code of a given process whose execution on the cloud would make the execution of overall process faster. The application is modeled as a control flow graph. The support of programmer is required to components annotations as the application functionality customization is difficult for ordinary user. The CloneCloud provides a simple approach to synchronize the target SMD and the cloud.

A mobile based framework is suggested in [9], to capture the social behavior of users in their working environment. The social behavior is captured using mobile phone sensors in the form of samples. The required information is then inferred by processing these samples. The limitations of mobile phones do not allow the large data captured to be processed locally. So, decision is made whether to process the samples locally or remotely. This decision is made by analysis the parameters of energy, data traffic and network latency. Real life implementation is shown using mobile phone Nokia 6120 as a client device and a server device based on Intel Xeon. The approach seamlessly processes the large data and able to infer the information required.

The Phone2cloud framework is suggested in [10], which is a complete computation offloading based system. A full quantitative analysis is done statically and decision is made whether to offload the computation or not. One important parameter is considered in this prototype which is user’s delay tolerance threshold. The prototype is implemented on Android and Hadoop environment and decision engine is built in order to analyze the energy consumption due to offloading. The average execution time of the application running on the SMD is compared with the user’s delay-tolerance threshold. If user’s delay is larger than computation is offloaded to the cloud, otherwise computation is made to run on the SMD. Before running, the power
consumption to run the application on the SMD ($E_{local}$) is calculated and compared with power consumption required to run the same application on the cloud ($E_{cloud}$). If $E_{cloud}$ > $E_{local}$ then computation is offloaded, otherwise it is made to run locally on the SMD. The main objective is to reduce the execution time and energy consumption cost in order to boost the application’s performance. In this way, the user’s experience is also improved.

A static framework is suggested in [11] which is COSMOS and provides computation offloading as a service for mobile devices. It receives demands for computation offloading from mobile users and then allocates them to a shared set of compute resources and then these resources can be acquired dynamically from a commercial cloud service provider. The framework provides benefits of computation offloading with minimum compute resource leasing cost. The applications like face detection and voice recognition can run seamlessly using COSMOS.

WORG (Weighted object relation graph) is suggested in [12] which involves static partitioning of application’s code. The optimize application partitioning is done in WORG by performing static analysis and dynamic profiling. The weighted object relation graph is constructed to depict the objects and relations between them. The most important parameter is bandwidth which is used with WORG to partition the application and decide which part of the application must execute locally on the SMD and which must execute remotely on the cloud. Basically, WORG uses branch and bound algorithm for partitioning of small application and uses min-cut based approach to partition large scale application.

A decision making mechanism is suggested in [13]. The decision whether to offload the computation to the cloud server or not is taken in this framework. The framework statically checks the parameters like memory capacity, processing power and battery life of the SMD. If the SMD successfully determines the value for these parameters the application is made to run locally on the SMD. Otherwise, the same application is made to run remotely within femtocell under which SMD is registered. Queue Assignment algorithm is used to run the application based on the computer time and deadline to execute the application. Basically, if the offloading of the application results in lower power consumption, only then it is offloaded otherwise application must be executed locally on the mobile device itself. The computation is offloaded using secure, low cost and low power Home Node Base Station having range 10-20 m approx. called Femtocell. For secure data transmission and secure cloud access, a process of bio-metric authentication is used. Significant amount of power saving is achieved using the femto-cloud architecture.

### 1.2 Dynamic offloading

Offloading decision is dynamic, when the program is partitioned at run-time and can adjust to varied run-time conditions, such as network latency or bandwidth. The dynamic offloading strategies initially statically analyze the code and instrumentation as well as parameters in order to perform dynamic/online profiling during execution. Based on the information obtained from dynamic profiling, the application is partitioned into client and server tasks. The execution then continues with the updated configuration. Dynamic approaches may use estimation techniques for optimize decision making. Following are some energy efficient frameworks used for computation offloading in MCC [14, 15, 16, 17, 18, 19, 20, 21, 22 and 23].

In [14] application partitioning is performed by analyzing the code. This framework is dynamic in nature. The potential benefits to run the specific application as a remote task are examined. Main purpose is low power. So, the tasks that are safe if made to execute remotely are identified. Also, the two versions of same task are generated in order to execute locally and remotely. This benefit of this technique is realized by implementing in the SUIF2 compiler, the result shows 13 to 15 times speed up in the face recognition code on iPAQ mobile applications. Hence, performance is improved.

In [15] an approach is suggested which decides which modules or components of Java programs should be offloaded. The technique first separates a Java program into methods and uses input parameters such as size of methods in order to compute execution costs for these methods. In order to make the optimal execution decision, this approach compares the cost of executing each method locally with the cost of executing that method remotely based on the status of the current wireless channel condition. But in [16], a parametric analysis is performed for partitioning the application. The computation and communication costs of application are analyzed. The optimal application partitioning is modeled as a min-cut network flow problem. Division of program into sub tasks in the form of modules is done and made to run on the mobile device and
on the server node. Depending upon the run time parameters value, the modules are assigned to the mobile device and server. A cost analysis of the parameters like, communication, data registration, computation and task scheduling costs is also done. The analysis helps to frame the partitioning as a single-source and single-sink min-cut network flow problem. This approach affects the performance of various applications like encode and decode from Minbenchand Mediabenchbenchmarks.

In [17] an energy-optimal application partitioning algorithm is suggested, which combines resource model by taken into account the overhead imposed by execution time and energy parameters in switching to runtime mode. An optimized software partitioning is done during compilation by making it an Integer linear problem. The problem of task dependencies is also addressed. The algorithm is useful for loosely coupled multi-processor systems. The network latency and energy consumption during communication is not considered. Hence, it is an in-efficient technique for applications requiring fast communication.

In [18], an approach is suggested which provides a dynamic offloading technique taking into account the behavior of application execution. The whole history of the execution pattern is profiled which is later used for taking optimal computation offloading decision. The computation offloading decision is made for each resource that is static (offloads most used classes), dynamic (moves only invoked classes), no action, or profile for later use. The offloading technique is useful for applications with large running times. The application executes faster using this technique.

A complete framework for computation offloading is suggested in [19] for Android called as Cuckoo. It provides a simple framework having a resource manager application for smartphone users, a runtime system and a programming model for developers which converges into the Eclipse build system. It allows local and remote execution by bundling both local and remote code into a single package. It uses QR code to make the remote resource known to the phone. IBIS communication middleware is used: Wi-Fi, Cellular or Bluetooth. The runtime system allows customization of remote implementations according to the availability of remote resources and the configuration can be changed in order to minimize energy usage and maximize computation speed.

MAUI (Memory Arithmetic Unit and Interface) [20] partitions the code for offloading to server dynamically and create two versions of mobile application using code portability, one version can be executed locally on the mobile devices and the other can be executed remotely in the cloud. The MAUI profiler component assesses the method each time it is called to check whether it saves energy which utilizes more processing and battery. MAUI build a linear programming formulation by taking into account three parameters at run time which are network bandwidth and latency, energy usage cost of mobile and communication cost. Based on formulation, MAUI is able to make the optimal application partitioning decision. Energy utilization is optimized as MAUI provides static fine-grained code offloading technique and hence fewer burdens are imposed on the programmer.

In [21], a decision engine is suggested which incorporates a fuzzy logic model by taking mobile and cloud variables into the decision making process. Notification services are used to introduce cloud parameters and rules from cloud to mobile. A neuronal algorithm is used to generate the rules dynamically in the cloud by analyzing the traces of code offloading. Fuzzy sets from mobile perspective are parameters such as bandwidth and data transferred. Cloud and Code traces are CPU instance and video execution. Various rules are considered to take decision whether to process the code locally or remotely.

A Lightweight Distributed Framework is suggested in [22]. The main objective is to minimize the use of additional resources in computational offloading for SMDs to cloud server. The features of framework are centralized monitoring and on demand access services of cloud’s resources for computational offloading. The framework allows mobile apps to operate in two modes: Online and Offline. Offline mode enables local execution and online mode allows offloading the computation to the cloud resources on demand basis. The SaaS model is employed in the framework for accessing the services of cloud server. The framework is evaluated by testing the prototype in the real MCC environment. The turnaround time, size of data to be transmitted and execution cost of the application evaluated are reduced.

An Energy Efficient Computation Offloading Framework (EECOF) [23] which is extension of [22] but is more energy efficient. EECOF provides lightweight distributed architecture and take advantage of SaaS model
in order to configure the intensive components on the node of cloud server and the IaaS model in order to adapt the offloading process of mobile applications. The cost of migrating application binary code and the active data state to the cloud server is reduced. EECOF framework allows mobile applications to run in two modes: offline and online. Offline mode allows mobile application to run on the local device and indicate that sufficient computing resources are available to run the application. The online mode has two operating procedures: Primary Operating Procedure (POP) which allows mobile devices to access the preconfigured services of the server of EECOF. The second is, Secondary Operating Procedure (SOP) which saves the data states of the running application and offloads the computation to the remote server. EECOF is evaluated different scenarios. The results are compared and it is found that the size of data to be transmitted and energy depletion cost as well as cost of runtime migration is reduced for computational offloading for MCC.

The comparison of all the discussed static as well as dynamic frameworks is given in Table 1.

<table>
<thead>
<tr>
<th>Year</th>
<th>Paper</th>
<th>Decision</th>
<th>Core Component</th>
<th>Parameters Considered</th>
<th>Supported Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>[2]</td>
<td>Static</td>
<td>Cost graph</td>
<td>Computation and data transfer time</td>
<td>Media bench and gnugo</td>
</tr>
<tr>
<td>2002</td>
<td>[3]</td>
<td>Static</td>
<td>Execution profile</td>
<td>Communication cost and connectivity of nodes</td>
<td>Text editor, Biomer and Voxel</td>
</tr>
<tr>
<td>2004</td>
<td>[15]</td>
<td>Dynamic</td>
<td>Application code</td>
<td>Data transfer</td>
<td>Java enabled applications and multimedia applications</td>
</tr>
<tr>
<td>2009</td>
<td>[7]</td>
<td>Static</td>
<td>Data flow graph</td>
<td>Network bandwidth and CPU load</td>
<td>High-rate data processing applications</td>
</tr>
<tr>
<td>2011</td>
<td>[8]</td>
<td>Static</td>
<td>Execution profiles</td>
<td>Computation cost and migration cost</td>
<td>Virus scanning and image search, video indexing, speech recognition</td>
</tr>
<tr>
<td>2013</td>
<td>[10]</td>
<td>Static</td>
<td>Prediction of User’s delay tolerance threshold</td>
<td>User’s delay-tolerance threshold, bandwidth, execution time, energy usage</td>
<td>Object recognition and general applications like sort, search and count</td>
</tr>
<tr>
<td>2015</td>
<td>[13]</td>
<td>Static</td>
<td>SaaS and IaaS model leveraging</td>
<td>Energy efficiency, run time migrating cost</td>
<td>General applications like sorting, searching, matrix operations or any computational intensive application</td>
</tr>
</tbody>
</table>
CONCLUSION
This paper provides a comparative review of various computation offloading frameworks for mobile cloud computing. The main objective of all these frameworks is energy saving which is most essential requirement of SMDs. The limitations of SMDs like slow execution speed, low processing power and less battery life can be addressed by offloading the energy intensive components of mobile applications to the cloud environment. The frameworks are classified based on the offloading decision taken i.e. static or dynamic, the core component used in framework which facilitates offloading, the various parameters which are analyzed before or during offloading and various real life applications that can run on these frameworks.

REFERENCES


