ENERGY EFFICIENT COMPUTING FOR SMART PHONES IN CLOUD ASSISTED ENVIRONMENT

Nancy Arya[1] Sunita Choudhary[1] and S.Taruna[2]

[1] Department of Computer Science, Banasthali Vidyapith, Rajasthan, India
[2] Department of Computer Science, JK Lakshmipat University, Rajasthan, India

ABSTRACT

In recent years, the employment of smart mobile phones has increased enormously and are concerned as an area of human life. Smartphones are capable to support immense range of complicated and intensive applications results shortened power capability and fewer performance. Mobile cloud computing is the newly rising paradigm integrates the features of cloud computing and mobile computing to beat the constraints of mobile devices. Mobile cloud computing employs computational offloading that migrates the computations from mobile devices to remote servers. In this paper, a novel model is proposed for dynamic task offloading to attain the energy optimization and better performance for mobile applications in the cloud environment. The paper proposed an optimum offloading algorithm by introducing new criteria such as benchmarking for offloading decision making. It also supports the concept of partitioning to divide the computing problem into various sub-problems. These sub-problems can be executed parallelly on mobile device and cloud. Performance evaluation results proved that the proposed model can reduce around 20% to 53% energy for low complexity problems and up to 98% for high complexity problems.

KEYWORDS

Mobile Cloud Computing, Mobile Computing, Computational Offloading, Dynamic Task Offloading, Energy Optimization.

1. INTRODUCTION

The revolution in technology has led to the number of individuals using smart mobile phones. These smartphones are not only used as a mode for communication but also provides support for complex applications that needs a high computational power. These applications can be offline or web-based. Power saving, high computation capability and storage capacity are the most desired features of a smartphone [13].

Mobile cloud computing provides a framework for portable applications where preparation and capacity of information are moved from mobile phones to remote servers. It is the technology that incorporates the concept of distributed computing with the high features to overcome the challenges related to mobile phones such as performance, environment and security [11]. According to the report of Allied Business Intelligence (ABI), up to 240 million corporations utilized the services of cloud through smart phones in 2015 that push the business of smart mobile phones to $5.2 billion. The constraints of mobile phones such as processing power, bandwidth, and limited storage can overcome with the usage of services of cloud computing. Now, smart phones are efficient to process a huge range of resource demanding applications which evacuate the power swiftly. According to Nokia survey report, the power capacity of smart

phones is one of the primary concern for customers. It is the restricting factor in the development and deployment of portable applications. This also restricts the phones to process computational applications. The hardware components such as bigger displays, CPU, Bluetooth, memory, GPS and network technologies such as 3G, 4G and Wi-Fi also need higher power. The energy consumption of any application is the sum of the consumed energy by several hardware components at the time of execution of the application.

To overcome these issues, computational offloading provides a better concept which transmitted the computational data to available cloud servers that have rich resources for computation and then process and return the results to the client [2],[3]. This approach can speed up the processing of computation as well as optimize the energy of power-limited devices. However, the migration of intensive tasks to the remote servers is a popular concept since the computer networks were developed.

The decision of computational offloading that which part requires to be offload based on certain parameters like network bandwidth, application type, size of application, architecture and storage limit etc. [7]. Offloading can be of static type in which the parameters of decision are defined at the time of development as well as of dynamic type in which parameters are defined at run time. Mobile computation offloading comes from the area of networking technologies such as Wi-Fi, 3G and 4G. However, these technologies enable offloading but consumes lot of energy in connecting.

Several offloading techniques and models exist to enhance the execution capability and energy optimization in mobile cloud environment but each of these has their own advantages and limitations. Conventional offloading frameworks used adaptive algorithms that transfer serious computations to the remote servers. These frameworks employ different levels for offloading applications at runtime but it includes migration cost of the computational components of the mobile application. Many state-of-the-art frameworks employ profiling and partitioning method that increases the overall execution time and energy consumption of local devices. Various frameworks do not focus on the additional overhead of partitioning and run time offloading. However, these frameworks emphasized on limiting the overheads of offloading and ensure to maintain the overall benefit of offloading.

The paper is emphasized on novel proposed framework for dynamic task offloading to achieve energy optimization and augment the performance. To achieve this, it proposed a suitable algorithm for optimum offloading by introducing new criteria and parameters in offloading decision making. It also introduces a partitioning concept to divide the computing problem into various subproblems. These subproblems can be executed parallelly on local mobile devices and cloud.

The main objective of this framework is to find the best optimum solution to offload intensive applications with proper utilization of assets so that the energy efficiency can be augmented. The proposed framework focuses on the decision that which parts of the application should offload, how they offloaded and where to offload. It also focused on to reduce the increased overhead of runtime component migration.

The experiment is performed in different phases: (I) Identification of problem and data generation. (II) Design and development of experiments for framework verification (III) Experiments will carried out. (IV) Validation and optimization of the framework. A set of experiments incudes two types of computational algorithms for different size of data to achieve

the efficiency in the proposed framework. The selected algorithms are merge sort and quick sort (sorting algorithms) as these are problem of low time complexity and matrix multiplication, the problem of high time complexity. However, in the case of merge sort and quick sort, the impact is small for small size data and is highly visible for a large size data. In the case of matrix multiplication, the impact is much visible even for the smaller size data. Analysis proved that the proposed framework achieved up to 20% energy efficiency for merge sort, around 53% energy efficiency for quicksort and up to 98% energy efficiency for high complexity problem such as matrix multiplication.

The paper is organized as follows - Section II describes the related work for offloading in mobile cloud computing. Section III discussed about the proposed framework, algorithm, its features and experimental setup. Section IV discussed about the results and performance evaluation and finally, Section V presents the conclusion and future scope.

2. RELATED WORK

Mobile cloud computing provides an infrastructure to extend the assets of the mobile devices over the network for mobile users and network operators [20]. It is the integration of various technologies such as mobile computing, cloud computing and wireless networks. Author described in [4] the various advantages of combination of the mobile computing and the cloud technology. However, mobile devices have several benefits, but also have some constraints. related to resources such as power, hardware components such as memory and storage, security, bandwidth and mobility [17]. Resource poverty is a main reason behind the issues related to Quality of Service (QoS) in mobile networks. These issues related to low bandwidth, unrealistic communication, heterogeneity and computation capability [24]. Mobile phones have limited battery capacity. The number of solutions has been proposed to optimize the energy by augmenting the efficiency of computation and proper management of the storage and the level of the display as described by [16], [23]. But these changes require modifications in hardware which increase the cost.

Offloading is a technique that migrates the time-consuming complex computations from the local device to the cloud. This reduces the energy consumption on mobile devices. Several of offloading frameworks [9],[25],[27] have emphasized on the efficiency of the process of offloading by including one or more parameters related to optimization such as computation time, energy consumption. According to [19], it is possible to save energy by migrating computations to the cloud server. But it is not necessary that an application takes less energy when it is migrated to the remote server. There are few other parameters that has to be consider while offloading the applications. These are energy overheads for reliability, security and data communications. An application should migrate only if the balance of energy is positive after considering all of these parameters. The Mathematical Arithmetic Unit and Interface (MAUI) architecture [10] has proposed for offloading to reduce the overall energy consumption for mobile gaming applications in cloud-based environment. This architecture reduces almost 27% of energy consumption on the mobile device. In [14], authors have proposed a dynamic algorithm based on lyapunov optimization for offloading computation to cloud. This algorithm decides which components of the software should be executed on cloud instead of local. This algorithm provides the low complexity for the solution of face recognition problems and saves 50% energy in simulation. Offloading also depends on the size of file. Authors have proposed an offloading algorithm [5] that depends on file sizes. The experiments of this work shown that there was no benefit in offloading of file size of up to 3 MB. However, for file sizes of 5 MB or more than 5

MB, there was advantage seen in energy efficiency as well as execution time. This is very effective model for taking decision for the offloading. In [18], authors proposed a framework for mobile devices known as Think Air for dynamic resource allocation and code offloading to the cloud server. It provides method-level offloading for computation and supports parallel execution of methods on different virtual machines. It also keeps track of energy for various hardware components. The model µcloud [8] is an energy saving model for mobile cloud computing. It is based on self-contained application components that are decoupled from each other. However, this model can execute only a single application partition at a time. In energy efficient multisite offloading algorithm [19], authors proposed a solution to visualize the range of tasks, tasks size, and dependency between these tasks of mobile application to increase the energy efficiency. Authors proposed [1] a framework for ultra-dense network environment to achieve the offloading for computational task. It is capable to reduce execution time by 20% and power consumption by 30%. Energy Efficient Computational Offloading Framework (EECOF) [26] proposed for the processing of complex mobile applications by using the cloud services with the migration of the minimum number of instances of application at runtime. The authors used bubble sort and matrix multiplication as an application to validate energy efficiency. This framework works on separate upload and download manager to synchronize the transmission and conserve the energy. The framework has tested with state-of-the-art frameworks and the results shows it is capable to reduce the size of data transmission around 84 % and power consumption by 69%.

MobiCloud [6] is a framework that supports Service-Oriented Architecture (SOA) for mobile ad-hoc networks. This framework considers every node as a service node in the network and creates replicas in the cloud. Authors proposed offloading [15] for mobile edge computing which is highly popular and useful concept for applications. In this concept, mobile devices can outsource their computations on the edges to save time and energy. It takes into account multiple tasks for each mobile client that is computation intensive. In [21], a framework has proposed for the adaptive partitioning and selective dynamic offloading for the selection of right category of cloud (central clouds and cloudlets) for which mobile applications should offloaded. The framework also works to minimize the execution cost during the decision of offloading. Authors proposed an algorithm in [30] for offloading of tasks in a heterogeneous mobile cloud environment. The algorithm addressed the problem of task offloading and task scheduling as a heterogeneous problem in this work. It is proposed to resolve the issues of energy utilization, time failure and execution cost for mobile devices. Compass-mobile framework [22] developed for high-performance applications in a distributed mobile cloud environment. The framework provides an energy-aware transparent model for developers to write code for regular android applications. These applications are then offloaded to the cloud as a computation. Authors have used merge sort as an application to validate the energy efficiency of the framework. Authors in [12] have proposed an algorithm that works for dynamic offloader to offload the computations from mobile to cloud environment. This offloader analyses the computations and migrates only intensive methods to the remote server to improve the energy efficiency. The algorithm has used applications such as quick sort and word count to validate the efficiency of the algorithm. They have used arrays of size from 100 to 100000 numbers and measured the computation time and power consumption for all these applications.

Authors proposed an offloading framework [29] for AHead-Of-Time (AHOT). This framework used the services of the cloud data centers to provide offloading as a service for mobile users. Experiments proved that this has been capable to reduce the execution time by 75% and energy consumption up to 76% for the number of applications. Authors used matrix multiplication of 1000x1000 for validating the criteria of an android-based offloading decision. In [28], the authors

proposed a mobile cloud environment offloading criteria depend on game theory which analyzes the effect of various parameters on offloading. The authors have focused on multisite risk offloading that takes the decision on the basis of risk evaluation. The decision criteria used matrix multiplication as an operation to decide the offloading.

3. PROPOSED FRAMEWORK

The research work is an attempt to proposed the framework to optimize the energy efficiency and execution time achieved during offloading by addressing new criteria in offloading decision making. Conventional frameworks processed cloud-based applications in three phases involving offloading initialization, computation offloading and application execution on cloud. In addition, the proposed framework also supported partitioning that divides the task equally on both mobile and cloud environments. As a result, overall energy consumption as well as execution time can be optimized. The main goal is the appropriate selection and offloading of computational tasks from the mobile environment to the cloud on the basis of their estimated energy and time. The framework employs an IaaS model (Infrastructure-as-a-Service) with SaaS (Software-as-a-Service) to reduce the energy consumption.

3.1. Architecture

The architecture of the proposed model consists of two components, mobile device environment, and cloud environment. The mobile device environment consists of any smart mobile device that can handle computing locally on the target operating system such as android, windows and i-phones. These devices are portable and constrained devices with limitations on memory, bandwidth, storage, power etc. As shown in figure 1, it consists of various components:

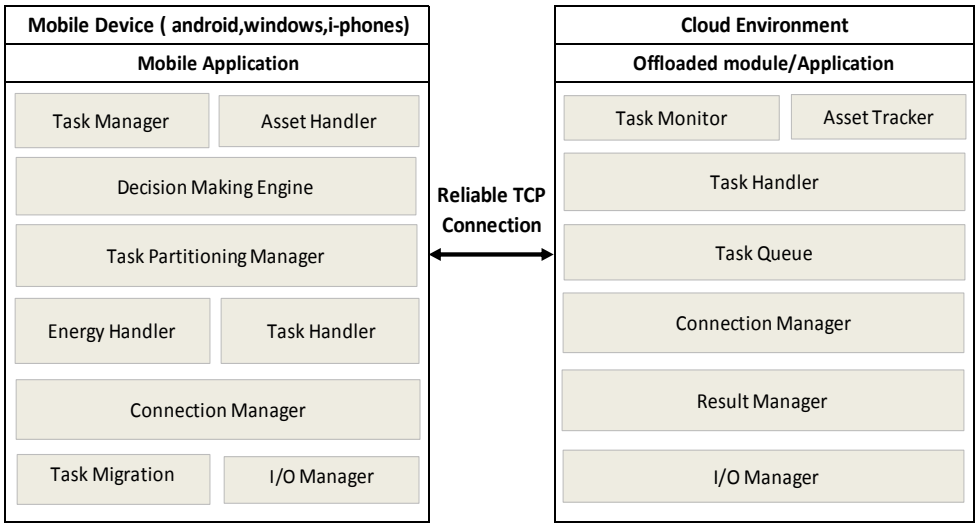


Figure 1. Architecture diagram of proposed framework

Task Manager is responsible to handle the computing problems requested by the user and put them in a queue in First-In-First-Out (FIFO) order. It takes the decision for benchmarking that calculates the sample values used to estimate the time and energy required for a computation. Asset Handler handles the resources required for computation and offloading such as memory and network resources. Decision Making Engine is the main control system of this environment

which takes the decision for offloading based on the input parameters and algorithm used to decide the appropriate location of the computation whether it executes on mobile or offload to the cloud. It interacts with other subcomponents to take the proper offloading decision. Task Partitioning Manager is responsible to examine the input problem and identifies those problems that can be divided into multiple sub problems and can be distributed across the local and cloud-based environments. After taking appropriate decisions, the sub-problems are offloaded to the decided locations for computing. The results are then combined to present a single output to the end user. Energy Handler is responsible to estimate the value of energy consumption for a given partition of problem. It provides this information to the decision-making engine to take the decision on optimum partitioning. Task Handler manages multiple operations like read, write and network connect etc. It also provides synchronization between multiple activities performed by the mobile client. Connection Manager on the mobile environment is responsible to manage the network connections and it is the control part for cloud environment. The used cloud services for computation is actually transparent to the end mobile user and it is managed by this component. Task Migration is the process that initiated after the decision of partitioning the problems into sub-problems. I/O Manager handles all the file and network input and output of mobile application.

On the other hand, the cloud environment, consists of remotely located computing resources. These resources are available on pay-as-you-go. The resources available on cloud are huge as it contains pool of large servers that can be allocated on demand. The cloud environment also consists of various subcomponents. Task Monitor is responsible to take care of all the process of problem migration across different locations and also handles the I/O functions. Asset Tracker is the module that is responsible to keep track of the resources required for various operations. Task Handler manages multiple tasks like read, write and network connect. It provides synchronization between multiple activities done by the cloud. Task Queue is responsible for receiving the problems from the mobile and scheduled them for execution. Connection Manager on the cloud side manages the connections between distributed locations. All the network connectivity is managed by the connection manager and is transparent to the other task execution engines. Result Manager handles all the results generated on cloud during execution of the computation problem and return these results to the mobile user to combined with the mobile device results. I/O manager is the important component of cloud that manages all the file and network input-output for the cloud application.

3.2 Proposed Algorithm

1. For a computational problem of size >10N, the benchmarking has done and average values are calculated. Calculate the time of computation, transmission time, time for reading and writing, energy consumption, energy for network transmission for unit amount of data. The unit time for mobile and cloud is calculated relative to the measured complexity of the algorithm. For example, if the complexity of the algorithm used during benchmarking is O(nlogn) and the total time for n no. of elements is t, then the unit time per element is t/nlogn. Benchmarking coefficients are also calculated for mobile and cloud environment for taking the decision of execution either on local or offload it to cloud completely or partially.

2. Inputs are available as a problem or set of problems $P = \{P_1, P_2, \ldots, Pn\}$.

3. The next step is the selection of computational problems. Selection is based on time complexity such as problem of low complexity, average complexity and high complexity. If the problem can divide into multiple sub problems is considered as non-monolithic in this algorithm and these subproblems executed simultaneously on mobile and cloud environments and then the obtained results with the respective environment are combined to produce final result. The size of subproblems would depend on the computation capacity of the mobile device which is obviously less than cloud.

   The concept of partitioning is helpful to improve the overall efficiency of the execution. Divide and Conquer strategy provides the best sorting complexity since they involve dividing the problems into subproblems and then compute the solutions to sub problems independently. In this work, merge sort and quick sort has considered as non-monolithic task to perform the experiment. The overall complexity of these algorithms is O(nlogn) which is efficient as compared to other sorting algorithms. If the problem is monolithic and cannot be divided into sub problems such as matrix multiplication, then the decision of executing the problem either completely on local environment or offloading it to the remote server. The time complexity of matrix multiplication is $O(n^3)$ which is higher as compared to sorting algorithms. However, matrices can be divided into subproblems using Strassen's Matrix multiplication algorithm but this algorithm reduces the complexity slightly.

4. The first parameter for taking decision for offloading is the time required for the execution.
   - The execution time for the mobile device consists of various parameters such as time consumed for reading data from memory $\boldsymbol{t_{mr}}$, time consumed for writing results back to memory $\boldsymbol{t_{mw}}$ and time for executing problem on mobile $\boldsymbol{t_{me}}$.
     Thus, the total time: $\boldsymbol{T_m = t_{mr} + t_{mw} + t_{me}}$
   - The execution time for offloading complete data on the cloud consists of additional parameters with the parameters of mobile device. These parameters are time consumed in network transmissions $\boldsymbol{t_n}$, time consumed on the cloud environment $\boldsymbol{t_{ce}}$ and time for combining the results $\boldsymbol{t_{mc}}$.
     Thus, total time for cloud: $\boldsymbol{T_c = T_m + t_n + t_{ce} + t_{mc}}$
   - The execution time in offloading the execution for both (y input data on mobile and N-y data on the remote environment) consists of additional time parameters needs for dividing the problem into sub problems and then combined together. These parameters are time consumed for dividing the problems into subproblems $\boldsymbol{t_{ms}}$, time consumed for the execution of subproblems on mobile $\boldsymbol{t_{mse}}$ and time consumed for subproblems on the cloud environment $\boldsymbol{t_{cse}}$.
     Hence, the total time for partitioning is: $\boldsymbol{T_{pa} = T_c + t_{ms} + t_{mse} + t_{cse}}$

5. Another parameter for taking decision for offloading is the energy required for the execution. If the execution time is equal for both of the environments (local and remote execution) then the environment with fewer energy requirements will consider.
   - The energy utilization for the mobile device consists of various parameters such as energy consumed for reading data $\boldsymbol{e_{mr}}$, energy consumed for writing results back to memory $\boldsymbol{e_{mw}}$ and energy for executing state $\boldsymbol{e_s}$.
     Thus, the total energy: $\boldsymbol{E_m = e_{mr} + e_{mw} + e_s}$

- The energy utilization for offloading complete data on the cloud consist of additional parameters with the parameters of the mobile device. These parameters are energy consumed in network transmissions $\boldsymbol{e_n}$. Thus, total energy: $\boldsymbol{E_c = E_m + e_n}$
- The energy involved in partitioning consists energy consumed for dividing the problems $\boldsymbol{e_{se}}$ and energy consumed for combined the results $\boldsymbol{e_j}$.
  Hence, the total energy consumed in partitioning is:$\boldsymbol{E_{pa} = E_c + e_{se} + e_j}$

6. In the case of non-monolithic tasks, calculate T_m, T_c, T_{pa}, E_m, E_c and E_{pa}. The partitioning decision in the form of O_a = {O_1, O_2,…,O_k) executes on local environment and O_b = {O_{k+1}, O_{k+2}, …… O_n), executes on the remote environment. The partitioning algorithm balance the problem to both the environments to achieve the simultaneous execution. According to the proposed partitioning algorithm, $\boldsymbol{O_a = N*B_c/(B_m+B_c)}$ where B_c is benchmark coefficients for cloud and B_m is benchmark coefficients for mobile and $\boldsymbol{O_b = N-O_a}$. The final result is the combination of outputs achieved from O_a and O_b. One of these sets may be empty.

7. In the case of monolithic tasks, the decision in the form of either execution on mobile or cloud. Calculate T_m, T_c, E_m, and E_c for monolithic task.

Start
// Case 1: Non-monolithic task
IF (P = non-monolithic)
{
Step 1: Calculate T_m, T_c, T_{pa}, E_m, E_c and E_{pa}
Step 2: IF (T_{pa}<= T_m)
{
Step 3: IF (T_{pa}<= T_c)
{
Step 4: Partition Task into O_a and O_b;
Execute O_a on local;
Execute O_b on cloud;
Join results;
Show results;
End;
}
Step 5: ELSE IF (E_c<E_m && E_c<E_{pa})
{
Execute on cloud;
Show results;
End;
}
ELSE {Goto Step 4;}
Step 6: ELSE IF(T_m<=T_c)
{
IF (E_m<E_c && E_m<E_{pa})
{
Execute the task on local;
Show results;
End;
}
ELSE {Goto Step 5;}
}
Step 7: ELSE {Goto Step 3;}

//Case 2: Monolithic Task
IF (P = monolithic)
{
Step 1: Calculate T_m, T_c, E_m, and E_c;
Step 2: IF (T_m <=T_c)
{
Step 3: IF(E_m<E_c)
{
Execute the task on local environment;
Show results;
End;
}
ELSE
{
Step 4:
Execute on cloud;
Show results;
End;
}
}
ELSE {Goto Step 4;}
End

8. The mechanism is used to achieve the optimization involves the sequence of operations of reading, write and network transmission in such a way as to facilitate the fastest execution of the task. The proposed framework works on parallel read and transmits operations to provide overall energy efficiency while executing the task. It also ensures that the computation on mobile and cloud environment should complete almost at the same time.

9. Figure 2 shows the sequence diagram. As the figure shows, once the partitioning is done, the input file starts to read and the transmission to the remote environment starts with the first read operation so that the simultaneous read and transmission operation takes place. when the transmission is completed, remaining data is computed by the local device. Once the execution on cloud completes, the results are then returned back to the local device for combined with the results on local device. Then the merged results are displayed to the mobile user.

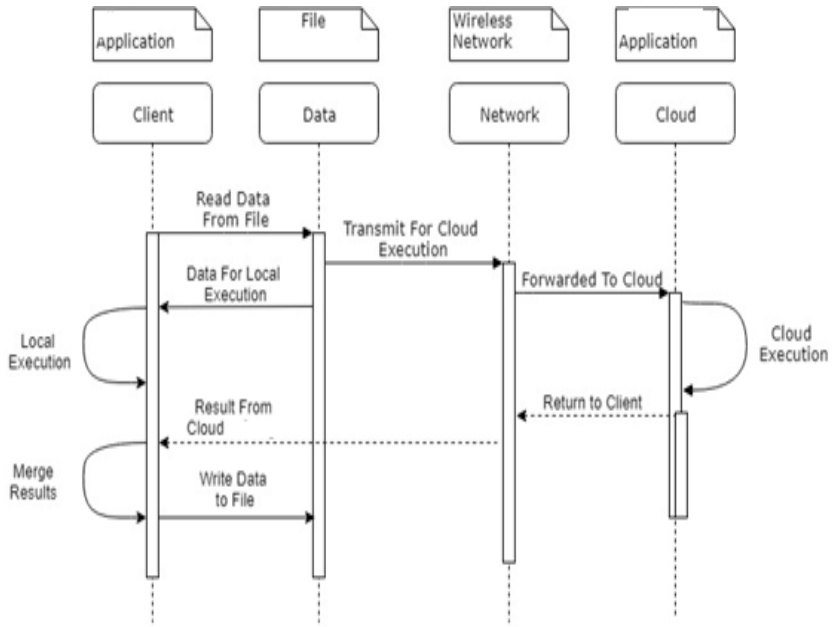


Figure 2. Sequence diagram for read, write and transmission operations

3.3 Experimental Setup

The proposed work includes experimental validation of the framework by involving the setup of mobile device mainly android smart phones and a cloud simulator on a virtual server. The experimental setup chosen for the proposed framework includes a mobile application running on an android smartphone Redmi note 4 with snapdragon 625 octa core, 4 GB RAM, 4100 mAh battery connected through a point to point TCP connection with the host intel core i5 system with 8 GB RAM running ubuntu operating system and docker on top of the operating system for simulating a virtual environment. Multiple instances of server code are running on top of the docker in order to simulate the cloud network.

The framework has been implemented on the mobile device and several computation problems are executed on the mobile device. The network connectivity between mobile device and the cloud network was through wireless connectivity. The mobile device runs the complex application used for partitioning, transfer it to the cloud, receive and merge the results. All the measurements related to time and energy are done by the mobile application.

This mobile application also runs the algorithm that takes the decision on offloading whether the application executes on mobile, offload to the cloud, or for a combination of both environments by taking various inputs from the user. The application then analyzes the data and take the decision. The decision depends on the estimated execution time and energy consumption of the application.

4. RESULTS AND DISCUSSION

The proposed framework is validated by designing the experiments and carried out to collect and analyze the data. The framework has evaluated the components of a mobile applications for mobile and cloud environment. An experiment is performed in three phases: (i) execution of computation on local device (ii) execution of computation on a cloud by offloading (iii) employing the proposed framework for computation. Energy efficiency is defined as an optimization of energy consumption. For real-time applications, execution time plays an important role to evaluate the performance. Hence, the energy efficiency and performance of the work has evaluated by comparing results at different experimental phases. To validate the energy efficiency, different sizes of data files are used. Execution time and energy are measured for all phases for mobile, cloud and for proposed framework. Initial measure has performed for merge sort. Results shows that offloading is useful to reduce the execution time, energy consumption and boost up the computation.

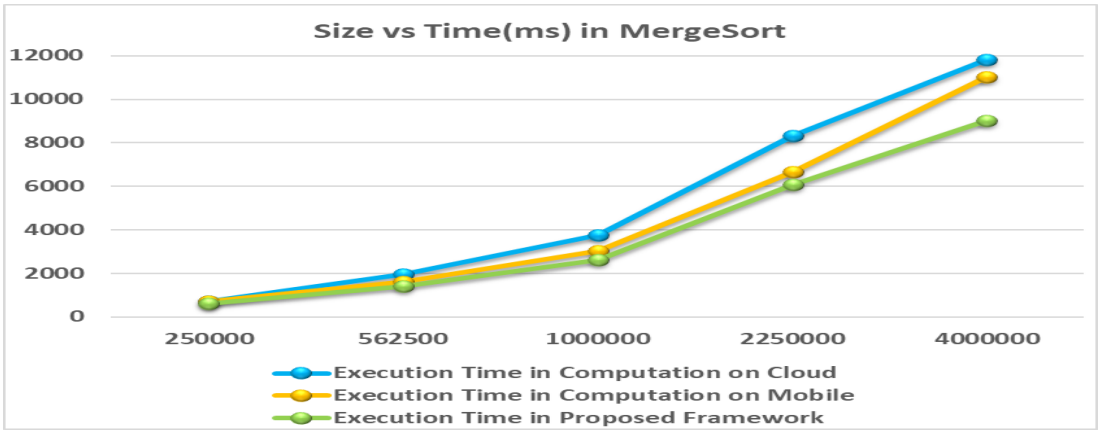


Figure 3. Size vs Time in merge sort

As shown in figure 3, for datasets of smaller size, results are almost the same for local as well as for cloud because of the significant overhead of network transmission for the merge operation. After analyzing the below figure 4, merge sort provides an advantage around 15% to 30% by employing the proposed framework as compared to the execution time taken by the cloud environment and this time involves the transmission time and time to receive results.

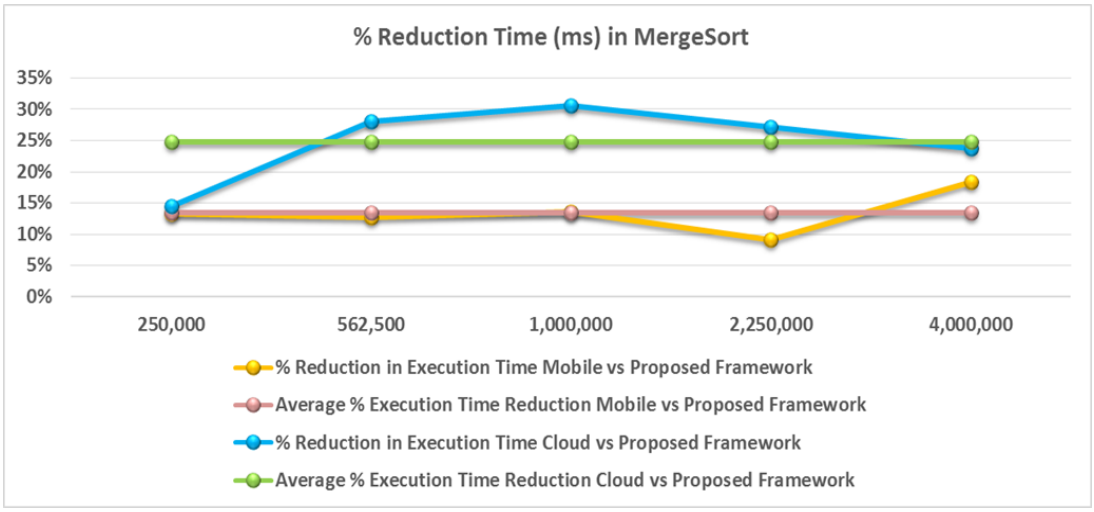


Figure 4. Reduction percentage in time for merge sort

Figure 5 shows the energy consumption for different data sizes for merge sort. As can be seen, the energy consumption also increases with the increment in data size. Merge sort is the problem of low time complexity of O(nlogn), so it performs well for small data also on the mobile device itself. But when data size increases, the network overhead of transmitting the data to cloud is balanced by the increased efficiency in computation on cloud. The average size for which the benefit of the cloud computation is visible at around 4000000. Thus, there is a definite advantage in parallel execution on mobile and cloud. It helps to optimize the overall energy consumption.

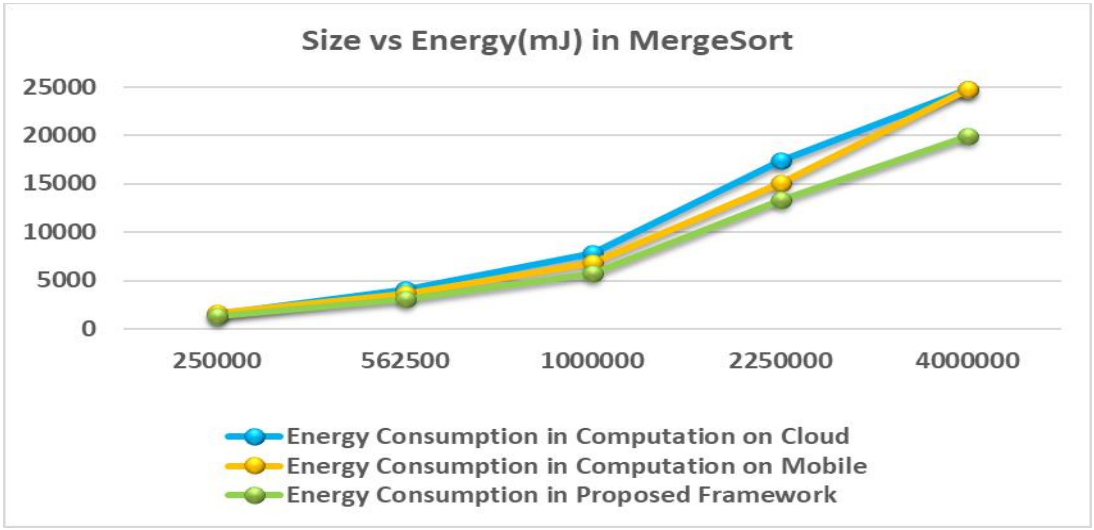


Figure 5. Size vs Energy in merge sort

In figure 6, energy consumption is around 17% to 20% when the proposed framework is compared with complete execution on local (mobile) and around 11% to 27% in the case of cloud for different size of data.

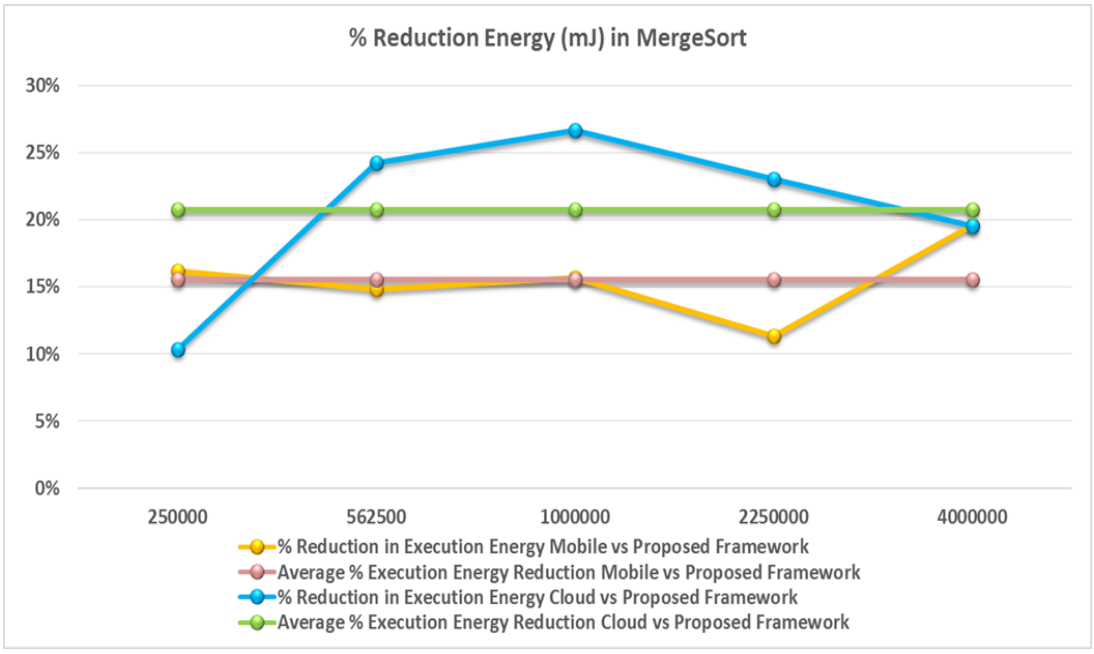


Figure 6. Reduction percentage in energy in merge sort

The second measure has performed for quick sort for the datasets of different size. As shown by figure 7, the execution time increases with data size in all the cases. Thus, there is an advantage only for larger data sizes of elements around 4000000 in offloading on cloud because the overhead of network transmission itself is too high for smaller datasets. Hence, the proposed framework gives benefit even for computation of small sizes.

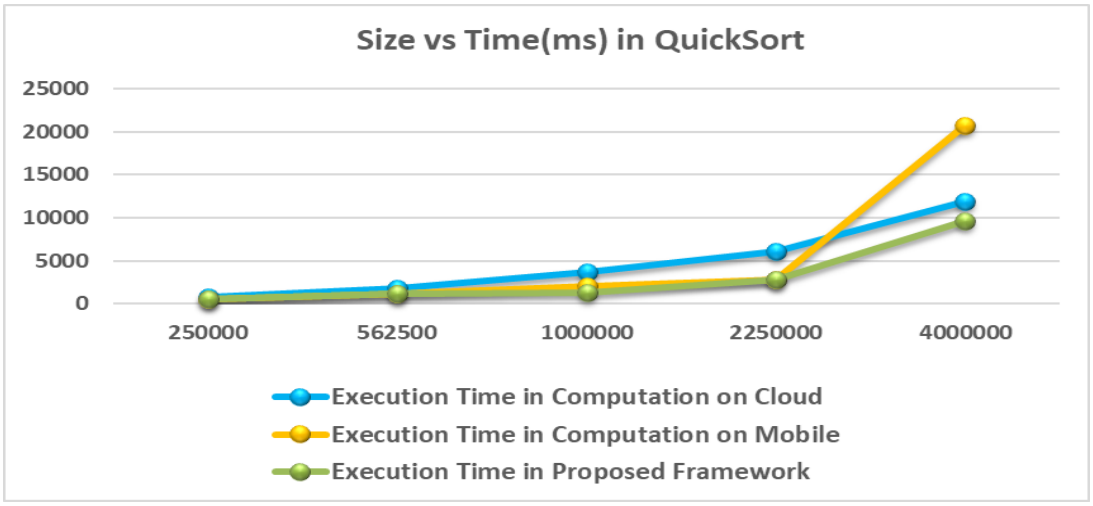


Figure 7. Size vs Time in quicksort

As by figure 7, in the local environment, this increment is very sharp as compared to a cloud environment which shows a less amount of increase. In figure 8, there is an advantage up to 54% in execution time when the computation is performed by proposed framework as compared to mobile. But, for the cloud environment, it lies around 20% to 66%.

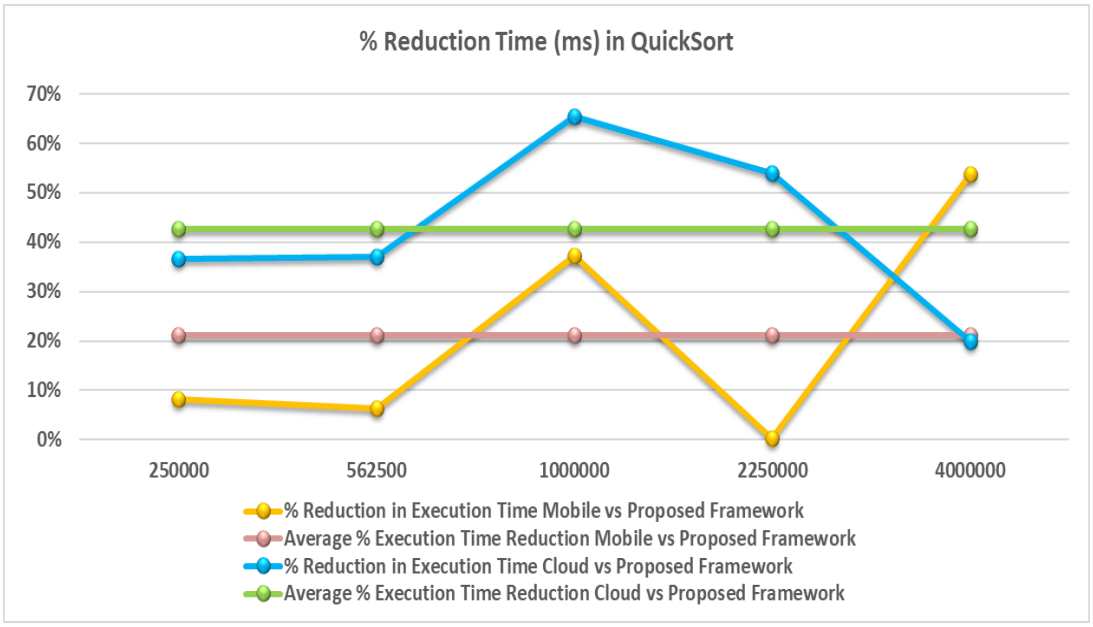


Figure 8. Reduction percentage in time in quick sort

As shown in figure 9, energy consumption increases with the size of computation. It can also be seen that the energy the consumption is high for large data sets on mobile and lowest in the case of the proposed framework.

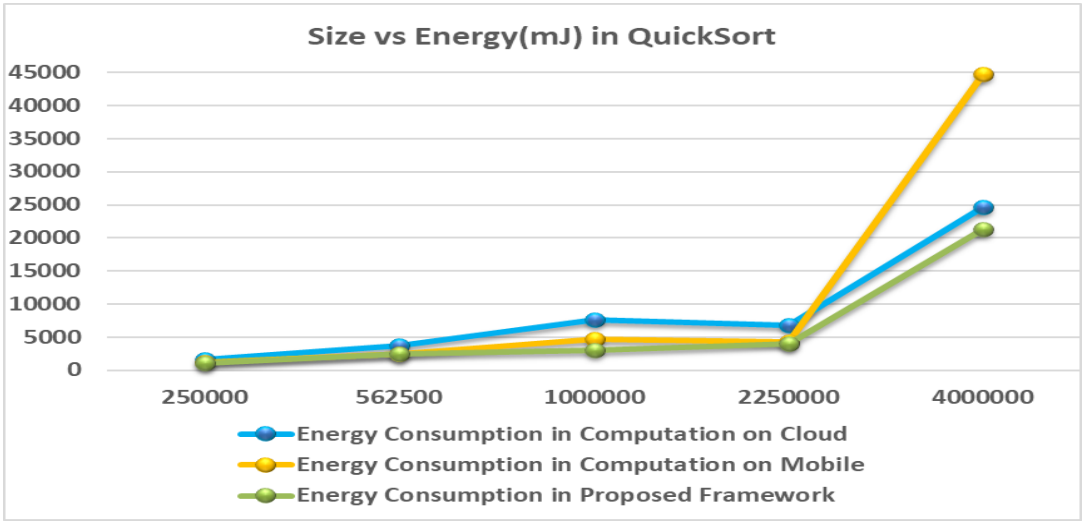


Figure 9. Size vs Energy in quicksort

If the energy is compared in figure 10, there is a benefit of around 1% to 52%, in the case of comparison with local execution and around 12% to 60%, when the proposed framework is compared with cloud computation.

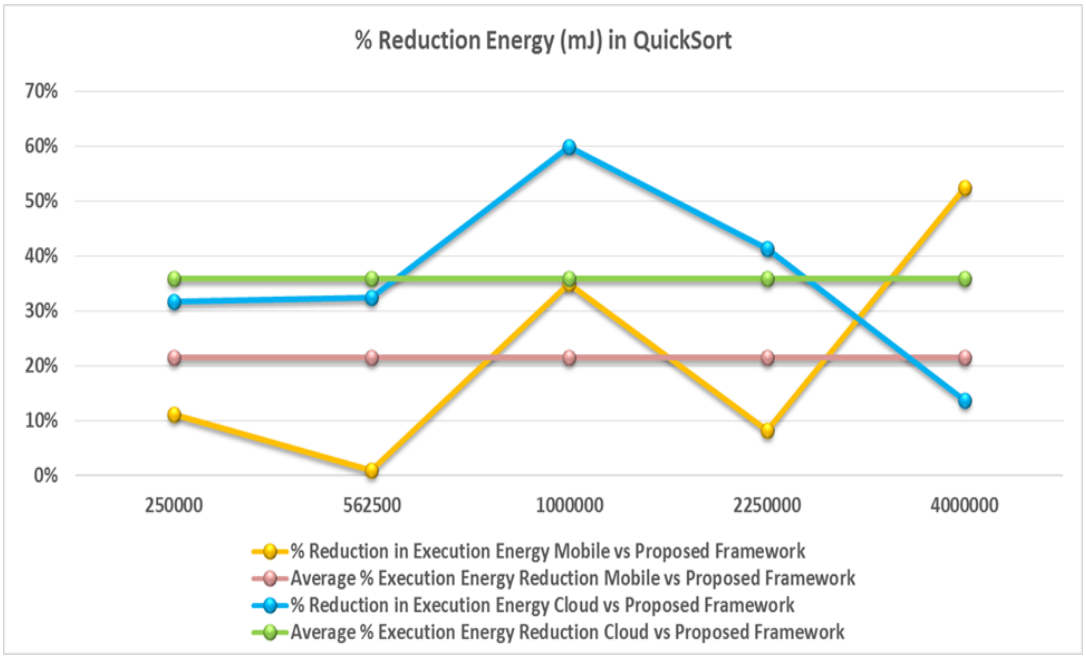
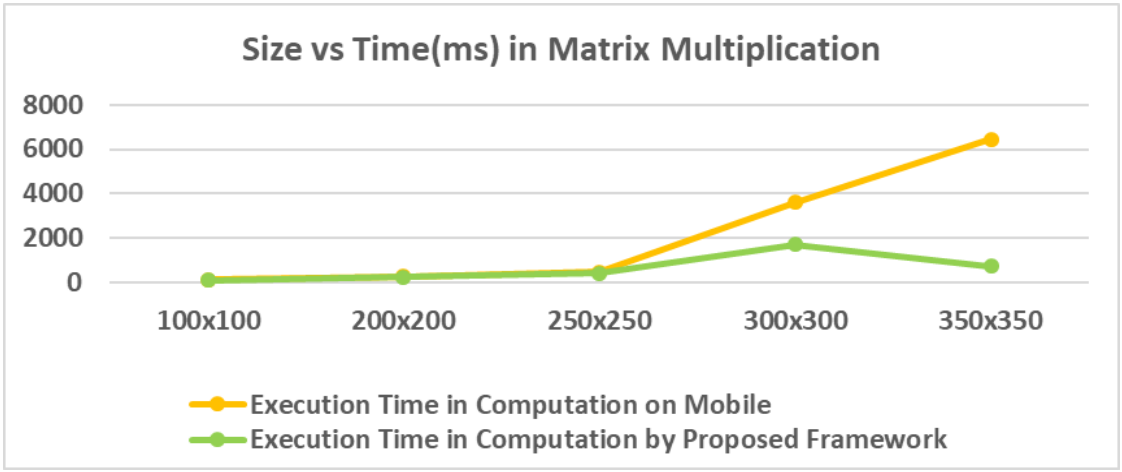


Figure 10. Reduction percentage in energy in quick sort

Next experiment has performed for matrix multiplication for the datasets of different size. As can be seen, the execution time for the computation increases with the increased data size. A distributed computation shows an immediate advantage for the small size of matrices also. Thus, matrix multiplication is the type of computational offloading which is quite advantageous in reducing the execution time for computation as well as energy consumption.



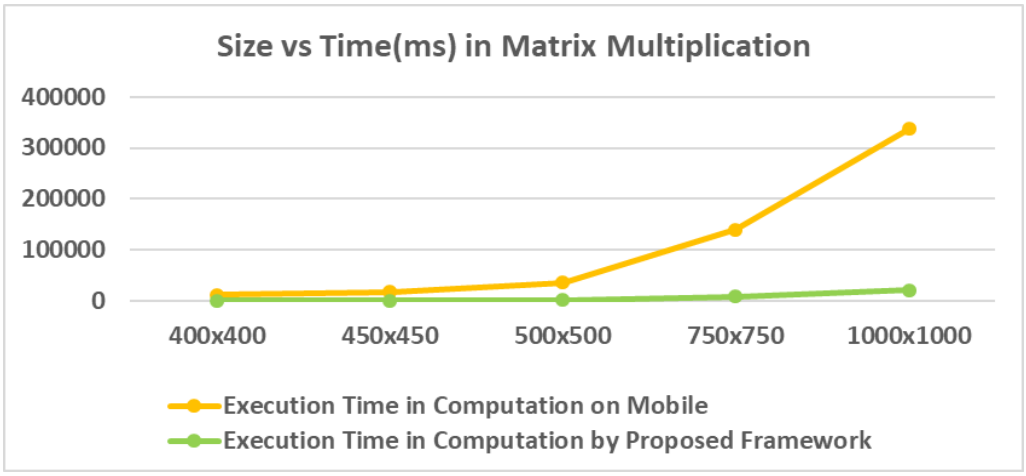


Figure 11. Size vs Time in matrix multiplication

A proposed framework gives instant benefit for the computation on small sizes also as the matrix multiplication is the problem of higher complexity $O(n^3)$. When time is considered for a very large matrix, there is a benefit at around 94%. In case of size vs time graph, the time increases exponentially with the increased data size as the complexity of execution dictates the energy consumption. As figure 12 shown, for a matrix of size 250x250, the reduction in execution time is around 13%. As the matrix size grows to 350x350, there is a significant advantage and for a matrix of 750x750 and beyond, the advantage is around 94% which shows the significant achievement.

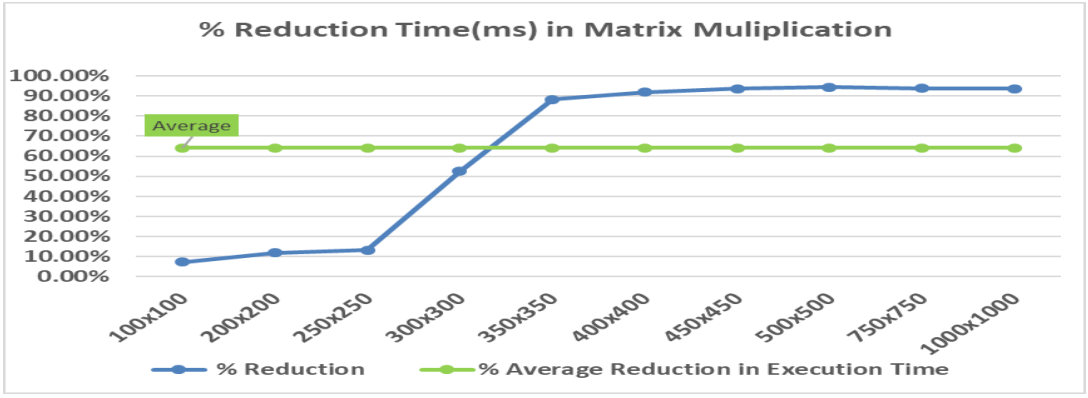


Figure 12. Reduction percentage in time in matrix multiplication

In the case of size vs energy graph, there is again a similar trend. The energy increases exponentially with data size as the complexity of execution dictates the energy consumed.

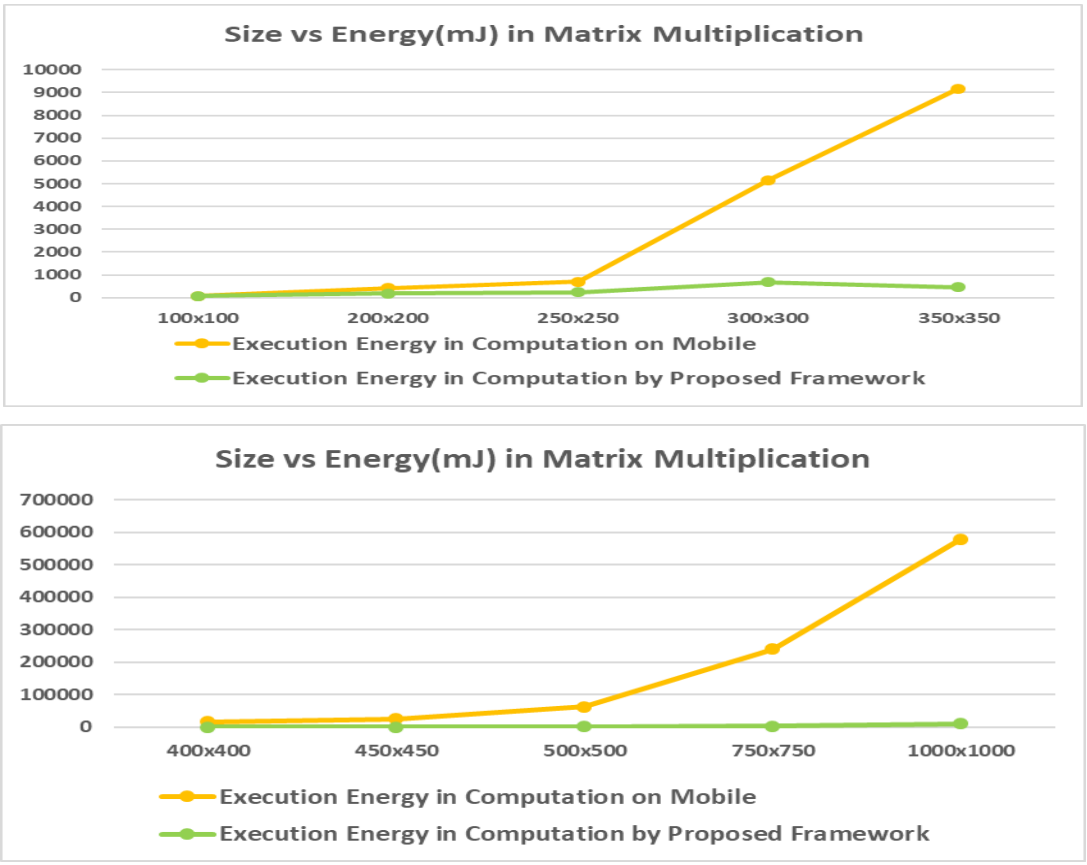


Figure 13. Size vs Energy in matrix multiplication

As shown by figure 14, energy difference increases initially for small data sets and saturates at around 95% which is an excellent gain. Beyond that, it saturates when it has reached the optimum value of around 98%.

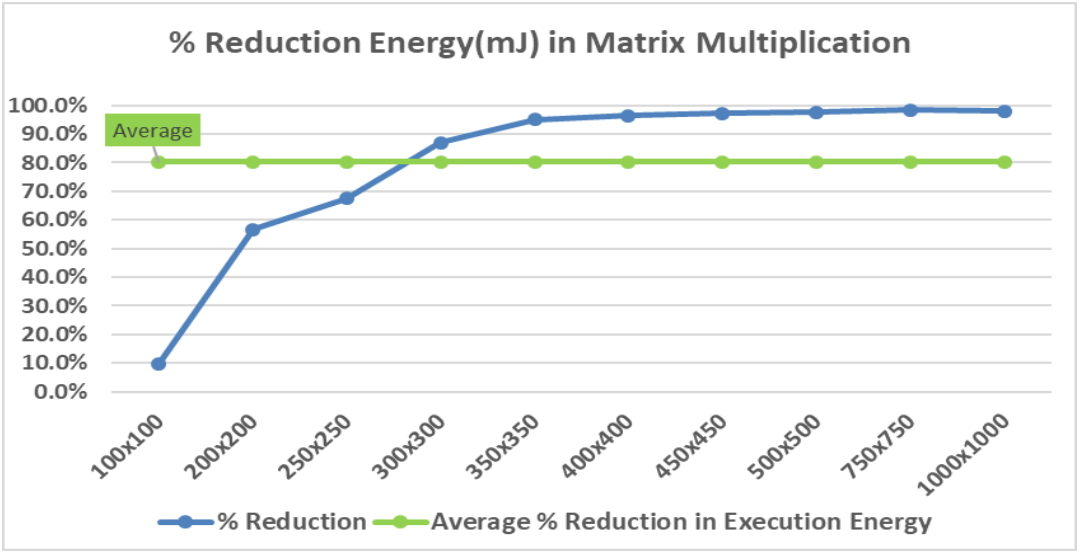


Figure 14. Reduction percentage in energy in matrix multiplication

As can be seen, the advantage that is obtained in energy by offloading the matrix to a cloud is small in case of small matrices 100x100. As the size grows, the advantage increases even more. For the matrix of size 250x250, the reduction in execution energy is around 67.5%. But as the matrix size grows to 350x350, there is a significant advantage and for a matrix of 750x750 and beyond, the advantage is around 98% which shows the higher efficiency.

5. CONCLUSION AND FUTURE SCOPE

Conventional frameworks focused on speed up the computation and saving energy on local devices. It plays an important role in the resource perspectives. In this paper, the author has proposed a novel offloading decision making framework to improve energy efficiency and performance. The framework used the concept of benchmarking of the computation before executing the problem and according to the average estimated valued of time and energy, the final decision for offloading has taken based on less energy consumption. The framework also used partitioning algorithm to divides the non-monolithic task across mobile and cloud environments and the simultaneous read and transmission operation has performed to optimize the overall energy as well as execution time.

Experimental results proved that the proposed framework worked well in improving performance and energy efficiency. The framework has verified by executing various algorithms as computational tasks to be migrated to the cloud. The selected computation tasks are non-monolithic task such as sorting (merge sort and quick sort) and monolithic task such as matrix multiplications. Due to the low time complexity, in the case of sorting, the achieved efficiency for smaller data size has not significant but as the size increases, the efficiency also increased. This is due to the overhead of sending the files on network are balanced against the achieved efficiency due to the offloading of large size of data. However, for large computations such as matrix multiplication, the observed energy consumption is around 98%, which shows that the efficiency of the proposed framework is higher.

Hence, the proposed combination of execution time and energy along with the algorithm of partitioning in the proposed framework has led to an efficient saving of energy for mobile environments. However, future work is possible by extending the scope of this research to explore the offloading for peer to peer networks. Edge clouds can be explored to optimize energy consumption as well as a frame work that works particularly for edge clouds can be evolved.

ACKNOWLEDGEMENTS

First of all, I hereby thanks to my supervisors for their expert guidance and cooperation. I am also grateful to the authors listed in the references to provide the valuable information.

REFERENCES

[1] Abdelminaam, Diaa Salama, et al. (2013), "Elastic framework for augmenting the performance of mobile applications using cloud computing", Proceedings of IEEE 9th International Computer Engineering Conference (ICENCO 2013), pp 134-141.

[2] Abdullah Gani & Han Qi (2012), "Research on Mobile Cloud Computing: Review, Trend and Perspectives", Proceedings of Digital Information and Communication Technology and it's Applications (DICTAP), IEEE Second International Conference, pp 195-202.

[3] Abdullah Gani, Ejaz Ahmed, Rajkumar Buyya, Saeid Abolfazli & Zohreh Sanaei (2013), "Cloud-Based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges", IEEE Communication Survey & Tutorials, Vol.16, Issue 1, pp337-368.

[4] Ahmed & Ejaz (2015), "Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges." Journal of Network and Computer Applications, Vol. 52, pp 154-172.

[5] Aldmour, S. Yousef, M. Yaghi, S. Tapaswi, K. Pattanaik & M. Cole (2017), "New cloud offloading algorithm for better energy consumption and process time", International Journal of System Assurance Engineering and Management, Vol. 8, No. 2, pp 730-733.

[6] Ali Mushtaq, Jashni Zain & Gran Badshah (2015), "Mobile cloud computing & mobile's battery efficiency approaches: A Review", Journal of Theoretical and Applied Information Technology, Vol. 79 No.1, pp 153-175.

[7] Antti P. Miettinen & Jukka K. Nurminen (2010), "Energy efficiency of mobile clients in cloud computing", Proceedings of Hotcloud'10,2nd USENIX Conference on Hot topics in Cloud Computing, pp 0-4.

[8] Bu Sung Lee, Erwin Leonardi, George Goh, Markus Kirchberg, Verdi March & Yan Gu (2011), "µcloud: towards a new paradigm of rich mobile applications", Procedia Computer Science, Vol. 5, pp 618–624.

[9] Carroll, Aaron & Gernot Heiser (2010), "An Analysis of Power Consumption in a Smartphone.", Proceedings of USENIX annual technical conference, Vol. 14.

[10] Cuervo, Eduardo, et al (2010), "MAUI: making smartphones last longer with code offload", Proceedings of the 8th international conference on Mobile systems, applications, and services (ACMMobiSys 10), San Francisco, California, USA, pp 49-62.

[11] Dhammapal Tayade (2014), "Mobile Cloud Computing: Issues, Security, Advantages, Trends", International Journal of Computer Science &Information Technology, Vol. 5, pp 6635-6639.

[12] Elgendy, Ibrahim A., Mohamed El-kawkagy & Arabi Keshk (2014), "Improving the performance of mobile applications using cloud computing", Proceedings of9th IEEE International Conference on INFOrmatics and Systems (INFOS2014), pp109-115.

[13] Gran Badshah, Jasni Mohamed Zain, Mohammad Fadli Zolkipli & Mushtaq Ali (2014), "Mobile Cloud Computing & Mobile Battery Augmentation Techniques: A Survey", Proceedings of IEEE SCOReD 2014, Malaysia.

[14] Huang, Dong, Ping Wang & Dusit Niyato (2012),"A dynamic offloading algorithm for mobile computing", IEEE Transactions on Wireless Communications, Vol. 11, No. 6, pp 1991-1995.

[15] Jiang & Weiheng (2018),"Energy-delay-cost Tradeoff for Task Offloading in Imbalanced Edge Cloud Based Computing", arXiv preprint arXiv:1805.02006v1.

[16] Kaushik, Nitesh &Jitender Kumar Gaurav. "A literature survey on mobile cloud computing: open issues and future directions.", International Journal of Engineering and Computer Science, Vol. 3, No. 5.

[17] Khan, Amreen & Kamlkant Ahirwar (2011), "Mobile cloud computing as a future of mobile multimedia database.", International Journal of Computer Science and Communication Vol.2, No.1, pp 219-221.

[18] Kosta & Sokol (2012), "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading", Proceedings of IEEE INFOCOM 2012, pp 945-953.

[19] Kumar, Karthik & Yung-Hsiang Lu (2010), "Cloud computing for mobile users: Can offloading computation save energy?", IEEE Computer Society, Vol. 43, No. 4, pp 51-56.

[20] Liu, Leslie, Randy Moulic & Dennis Shea (2010), "Cloud service portal for mobile device management", Proceedings of e-Business Engineering (ICEBE), IEEE 7th International Conference.

[21] Liu, Xing, Songtao Guo & Yuanyuan Yang (2017), "Task Offloading with Execution Cost Minimization in Heterogeneous Mobile Cloud Computing", Proceedings of International Conference on Mobile Ad-Hoc and Sensor Networks, Springer, Singapore.

[22] Lordan, Francesc & Rosa M. Badia (2017), "Compss-mobile: Parallel programming for mobile cloud computing", Journal of Grid Computing, Vol.15, No. 3, pp 357-378.

[23] Mayo, Robert N.& Parthasarathy Ranganathan (2003), "Energy consumption in mobile devices: why future systems need requirements–aware energy scale-down", International Workshop on Power-Aware Computer Systems, Springer, Heidelberg.

[24] Rahimi, M. Reza, Nalini Venkata Subramanian& Athanasios V. Vasilakos (2013), "MuSIC: Mobility-aware optimal service allocation in mobile cloud computing", Proceedings of Cloud'13, IEEE Sixth International Conference on Cloud Computing, pp 75-82.

[25] Rifang Niu, Wenfang Song & Yong Liu (2013), "An energy efficient multisite offloading algorithm for mobile devices", International Journal of Distributed Sensor Networks, Vol. 9, Issue. 3.

[26] Shiraz Muhammad, Abdullah Gani, Azra Shamim, Suleman Khan & Raja Wasim Ahmad (2015), "Energy efficient computational offloading framework for mobile cloud computing", Journal of Grid Computing 13.1, pp 1-18.

[27] Smailagic, Asim & Matthew Ettus (2002), "System design and power optimization for mobile computers", Proceedings of VLSI, IEEE Computer Society Annual Symposium.

[28] Wu, Huijun & Dijiang Huang (2014), "Modeling multi-factor multi-site risk-based offloading for mobile cloud computing", Proceedings of 10th International Conference on Network and Service Management (CNSM).

[29] Yousafzai & Abdullah (2016), "Computational offloading mechanism for native and android runtime based mobile applications", Journal of Systems and Software, pp 28-39.

[30] Zhou & Bowen (2018),"An Online Algorithm for Task Offloading in Heterogeneous Mobile Clouds", ACM Transactions on Internet Technology (TOIT) 18.2.

BIOGRAPHY

Nancy Arya is a Ph.D. research scholar in Computer Science and Engineering. She has completed her M.Tech. in Computer Science and Engineering from Jagannath University, Jaipur in 2013 and MCA from Rajasthan Technical University, Kota in 2010. She has 5 years of experience of teaching and 1.5 years of industry. She has presented many papers in national and international conferences. She has published around 15 research papers in reputed journals and conferences. Her research interests includes Networking, Software Engineering and Cloud Computing.