Introduction

It can be anticipated that more and more applications are to be converted into mobile cloud computing (MCC) [1]. MCC is a framework which augmented the capabilities of mobile computing with rich re-source cloud computing. Another side, offloading is the technique which migrates the compute intensive tasks of application to the remote cloud while keeping the total cost as small as possible [2]. Many efforts have been made on computational offloading in mobile cloud computing environment to achieve their objectives such as energy efficient task assignment, minimize total energy consumption and vice versa [3].In this paper, we are studying the application partitioning and energy efficient task scheduler in mobile cloud environment. In real practice, a real time application such as face recognition is composed of tasks.Whether offload or not offload tasks while considering, network, device and workload status during application execution is remained challenged [4].

Energy Efficient Task Assignment has three techniques for the application execution inside mobile the device, for example full offloading where energy hungry tasks computationally offloaded to the surrogate server it could be local or remote cloud. In order to minimize total energy yanother approach non- offloading retained all tasks locally; extensive tasks consume much more energy of the device. However, we have proposed EETA (energy efficient task assignment) algorithm and that is a partial offloading technique so that minimizes and prolong the device energy as well as communication energy. EETA is the Minimization problem and well known as the NP-hard problem, it is not trivial to solve.However, the energy efficient task scheduler still facing a number of challenges as listed follows Application weighting: every application has different weight with respect to tasks. The application is composed of multi-size tasks. Energy efficient application partitioning and task assignment plays important role in performance. Previous, proposed scheme such as static analysis could not guarantee the optimal and energy efficient task assignment due to adaptively change in mobile device status and workload. A real time inference algorithm must be proposed to be able to tackle run time situation of mobile cloud application.

Network adoption: Application offloading to the cloud server introduces extra communication burden. The behavior of the network changes time to time. While, during peak hours due to high traffic most of network nodes are affected by congestion. A network adaptive type real time algorithm must be proposed which cope proceeding challenging.

Dynamic and Effective partitioning: fine grained, has multiple processing costs such as local execution and remote execution. Dynamic and effective assignment of tasks required real time algorithm.

Paper sections organized as follows section 2 tells about related work and motivation. Section 3 explains the system description. Section 4 tells about system model and problem formulation. Section 5 about simulation and results

2. Related Work

Offloading is a technical way to offload the energy-hungry tasks to the external surrogate servers; nevertheless, many efforts have been made in offloading Scheme to achieve multiple objectives under different cost models. Some of the applications including face-recognition, 3D Gaming and Augmented Reality, could take a significant amount of time due to their computationally intensive nature. Processors for mobile devices are gradually getting faster year by year; however, with-out aid from special purpose hardware, they may not be fast enough for those computationally intensive applications. In this section, we describe the most prominent solutions of mobile cloud computing (MCC) and high-light their main shortcomings. MCC is composed of the capabilities of mobile computing and cloud computing. Cloud computing brings mobile devices a great number of computing resources through virtualization technology. Computation offloading has great attention nowadays in regard academic as well as industry. Research literature has been made many efforts on the computation offloading with their respective objective (i.e., energy consumption, execution time minimization).

Could computing can offer computing source, and users can use these to decrease the amounts of computation on mobile systems and save energy. Thus, cloud computing can save energy for mobile users through computation offloading [5, 6]. However, migrate a large amount of data in a low-bandwidth network will cost much more energy than local computation. Offloading [7, 8, 9] is a method to offload part of the computation from the mobile device to another resource-rich platform A number of modern works intend different methodologies to offload computation for specific applications [10, 11 12].

In an additional part, a lot of works have a discussion about on how to pre-estimate the real boost in terms of energy [13]. For a code complication, offloading might consume more energy than that of local processing when the size of codes is small [14].The study [15] suggests a program partitioning based on the assessment of the energy consumption (communication energy and computation energy) before the program executes. The optimal program partitioning for offloading is calculated based on the trade-off between the communication and computation costs to do offloading in the dynamic environments, Tang et al. [16] consider three common environmental changes: power level, connection status, and bandwidth. But they just ex-plain the suitable solutions for offloading for different environments separately Chun et al. [17] present a system to partition an application in dynamic environments. The proposed system follows three steps with different requirements related to the application structuring, the partitioning choice, and the security.

3. Problem Description

3.1. Application Scenario

Elastic application clutches with several tasks in the course of data dependency. In Algorithm 1 EE-TA (Energy Efficient Task Assignment) the input is call graph G (V; E). However, V has a two disjoint subset of tasks such as Vl and Vc local disjoint set and cloud Disjoint set respectively. Whereas, E is the communication between caller and caller that could be Ptr between two tasks or two disjoint circles. EETA has a lot of variables, W is the disjoint set weight, R, mobile device available resources, D device current workload, F mobile service rate (i.e., processing power, velocity) and T P, RP is required power and total power of the cur-rent device.

3.2. System Model and problem Formulation

We analyze the application tasks as call graph as shown in Figure 1(A). Furthermore, application partitioned into consumption graph as shown in Figure 1(B) We show the application system as call graph G (V; E). However, V has a two disjoint subset of tasks such as Vl m and Vrc local disjoint set and cloud disjoint set respectively. Whereas, E is the communication between caller and called that could be Ptr between two tasks or two disjoint circles. EETA has a lot of variables, W is the disjoint set weight, R, mobile device available resources, D device current workload, F mobile service rate (i.e., processing power, velocity) and T P, RP is required power and total power of the current device.

3.3. Energy Model

Energy consumption of mobile devices depends on the computation and communication loads. To explore the energy consumption of each task, we suppose the task computation requires I instructions. The task needs to deal with D bytes of data and will generate D' bytes result, we use B to stand for current network bandwidth. It will task D /B seconds to transmit and receive data. We define our task as follows: Definition 1. [Application Task] T (I, D, D').

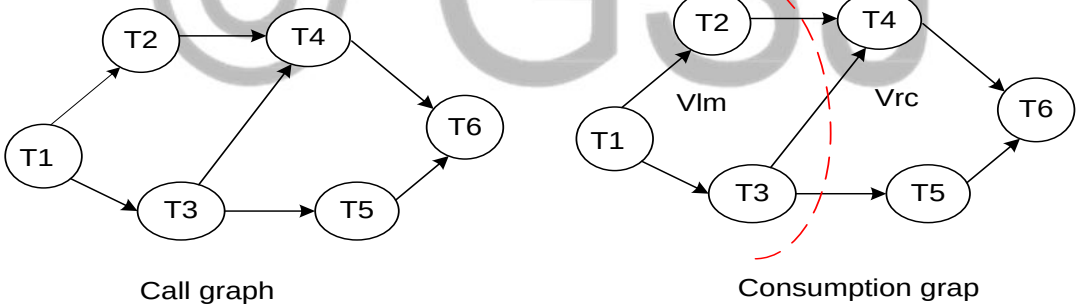

Figure1. Application Scenario

A mobile application task T has I instructions to be executed. The task uses D bytes input data and generates D' bytes output data. The mobile system consumesP_c (watt per instruction) for computing andP_{tr} (watt per second) for sending and receiving data. If we choose to execute our task using offloading, we need to send our code and data to the server. So the total energy consumption is:

$$E_{offloading} = P_{tr} * \frac{D}{B} \qquad (1)$$

Suppose the output data $D'$is k times smaller than the original data D, we use compress ratio k to describe the relationship between input data and output data: $(D' = D \in k)$. In order to simplify the compression algorithm impact on the size of output data when the size of input data is different, we just consider the application task with fixed compression ratio. If the mobile device performs the task, the energy consumption is:

$$E_{Non-offloading} = P_c * I + P_{tr} * \frac{D'}{B} \qquad (2)$$

3.4. Problem Formulation

The amount of energy saved is:

$$E_{save} = \frac{E_{offloading} - E_{Non-offloading}}{E_{offloading}} * 100\% \qquad (3)$$

$$E_{save} = \frac{E_{offloading} - E_{Non-offloading}}{E_{offloading}} * 100 \qquad (4)$$

$$P_{tr} * \frac{D}{B} - P_c * I - P_{tr} * \frac{D*k}{B} - P_c * I =$$

$$P_{tr} * \frac{D}{B} - P_c * I - P_{tr} * \frac{D'}{B} = P_{tr} = (1-k)\frac{D}{B} - P_c * I \qquad (5)$$

Theorem1. $\min_{\sum v \in V} T(V_l m, V_r c)$ Is the NP-hard and optimization problem?

Proof. Our proposed framework is following partial offloading and save energy as compared to full offloading such as $E_{save} = \frac{E_{local} - E_{total}}{E_{local}} * 100$. The final gain would be resultant of energy saving as compared to traditional techniques (i.e., full offloading and non -offloading).

Algorithm 1 EETA

Require: $G(V, E)$ Input call graph
Ensure: Optimize $E = \{V_l + V_c + P^{tr}\}$

```
1:  Declaration W; Net;R ;D ;Φ ;μ,TP,RP
2:  function PROFILING(E = {V_l + V_c + P^tr})
3:      if T < W and W < R then
4:          if RP ≤ TP and W < R then
5:              E ← R, Φ
6:      else
7:          APP Solving{Net, D, T}
8:          APP Partitioning{V_l, V_c} via Algorithm 2
9:          V_l ← R, Φ
10:         Partition Migration {V_c + P^tr}
11:         V_c ← R, μ follow Algorithm 3
12:     return E
```

In Algorithm 1, step-1 shows different variables for application partitioning and resource mapping. Step-2 profiles the application before the start via application task characteristics (i.e., weight, and required power). Steps-3-4-5 shows that if the requirement of the application is catered and resources are available, retained the application tasks locally. Step-7 makes offloading decision based available device power status, required power status, program weight, network status and current device workload before execution of the diligence.

Pace 8-11 make application partitioning according to the characteristics and mapping the resource to required disjoint sets (i.e., $\boldsymbol{V_l}$ retain locally and $\boldsymbol{V_c}$ offloaded to the remote cloud). Finally, step-12 return the energy efficient task assignment by looking at both de-vice and communication energy simultaneously. The time complexity is iteratively energy efficient and calculated via ($\boldsymbol{log}\ |\boldsymbol{V}\ |\boldsymbol{E}$). Algorithm 1 only works on partial offloading against either non-offloading or full offloading. Algorithm

Algorithm 2 Local Mapping Resource

Require: $V_l \in V$; $V_l \cup V$;
Ensure: Optimize $E(V_l \in V)$
1: Declaration W; R ;D;W;Φ RP ;
2: **function** MAPPING($E = \{V_l\}$)
3: **if** $E(V_l < W$ and $W < D)$ **then**
4: $E \leftarrow R, RP, \Phi$
5: **else**
6: $E(V_c \leftarrow R, \mu)$
7: **return** E

2, follow Algorithm 1, mapping the required power and memory (i.e., processor process and storage) for un-offloaded disjoint (e.g.,V_l) retained locally. Since, Steps 3-6, is the assignment process of tasks (i.e., un-offloaded) by perspective of energy and resource utilization. It iteratively chooses the tasks which is following precedence sequence order in the given environment. that could be adaptively change (i.e., device status and workload). The time complexity is polynomial and energy efficient (i.e., $O(N + N)^2$)

Algorithm 3, follow Algorithm 1, mapping the

Algorithm 3 Cloud Mapping Resource

Require: $V_c \in V$; $V_c \cup V$;
Ensure: Optimize $E(V_c \in V)$
1: Declaration W; R ;k;W;Th;μ ; B_{up}, B_{down}
2: **function** MAPPING($E = \{V_c + P^{tr}\}$)
3: **if** $E(V_c < W$ and $W < k)$ **then**
4: $E(V_c + P^{tr}) \leftarrow R, \mu$ follow FIFO order
5: **if** $P^{tr}(B_{up}, B_{down<Th})$ **then**
6: Re-Partitioning
7: **else**
8: $E(V_l \leftarrow RorV_c \leftarrow R, \mu)$
9: **return** E

Required power and memory (i.e., processor and storage capacity) for computationally offloaded disjoint (e.g.,V_c) migrated to the server k. Since, Steps 3-6, is the assignment process of tasks (i.e., offloaded) by perspective of energy (i.e., communication and process (queue and execution)) and resource utilization follows FIFO order. It iteratively chooses the tasks which are following precedence sequence order in the given environment that could be adaptively change (i.e., server status and workload). The time complexity is polynomial and energy efficient (i.e., $O(N + N)^2$).

4. Performance Evaluation

The mobile application holds several energy-hungry tasks in order to prolong the battery power and average energy utilization of the system proposed algorithm EETA has significant simulation results. In the meantime, it can adapt to runtime environment changes as compared to existing methods. To evaluate the EETA energy efficient algorithm we need to know three kinds calibration values such as: Unchanging standards: some set of parameters is fixed by the application developer during design, analysis such as power con-

servative elements are required to be fixed for different kinds of devices. Since parameters, calibration is calculated via Table 2. Table 2 has a set of fixed and adaptive values of the tasks and relatively for the system.

- **Explicit principles:** Speedup factor of cloud computing and mobile cannot be controlled over the time. Furthermore, bandwidth B=upload and download values vary due to environmental changes, set of tasks with different data size may suffer due to environmental changes and it could be reasoned for degrading QoS (quality of service) and performance
- **Premeditated standards:** These kinds of standards supposed to be changed by device Characteristics and input parameters of an application tasks Application program is calculated by program pro-file after the application has been started in the period of run time and synchronously adaptive. Corresponding performance is calculated based on catered schemes that could be depicted as an always non-offloadable and always offloadable. The consequences can be handled in the following way:
- **Always Non-Offloading:** This is the property and a characteristic of the application tasks since all computation happens inside the mobile device. This scheme might be harmful and less efficient since the limited resource device not able to tackle compute and energy-hungry tasks inside device results higher battery consumption and average performance simulated go down.
- **Always Full Offloading:** Cloning the image of tasks moved from mobile device to the remote cloud is the effective way as compared to Non-Offloading. It is directly proportional to the speed factor of the cloud and bandwidth corresponding to the network while adaptively variation in the speed factor and bandwidth the full offloading scheme shows bad performance and less efficient in the case of the result.

This scheme encompasses on both always Non-Offloading and always Full Offloading schemes, it could have better outcomes even though the deviation in speedup factor as well as wireless in excess of the point of time. Our proposed system mobilized the saved cost in the following way:

$$OSC = 1 - \frac{EETA}{Non - Offloading} * 100\% \qquad (6)$$

Whereas, OSC (Offloading save cost), in order that minimizes the total cost (i.e., mobile and communication energy) through application execution.

4.1. Task Energy calculation

An application program such as tasks (granularity could be a method or class or object) can be observed via program profiler. Even though, each task is required to power consumption and memory for the execution. It could be executed either on the local device with service rate ϕ_m or remote execution is followed by μ_m. However, a set of cloud servers can be represented by $k \in K$. Nevertheless, cloud servers are homogeneous in nature and have the same speed and storage capability.

4.2. Workload Analysis

We have examined four real-time application workload such as a Linpack math tool, 3D-Game EEG Beam 3D Game, Augmented Reality (a gesture application) and Face Recognition application. Each workload is characterized by some properties such as task size, memory requirements, and required power consumption execution time. The individual application has benchmark application features, each task is required to execute with a given threshold, it could be energy or execution time bound. We can be seen that Figure 2 and Figure 3 speedup factors F synthetically and synchronously, clutches are whipping influence over the energy consumption and application performance.In view of the fact that, proposed algorithm EETA has better consequences over traditional schemes i.e., full offloading and Non-Offloading. A number of tasks generated randomly, and arrival rates follow uniform distribution and it could be a Poisson process.

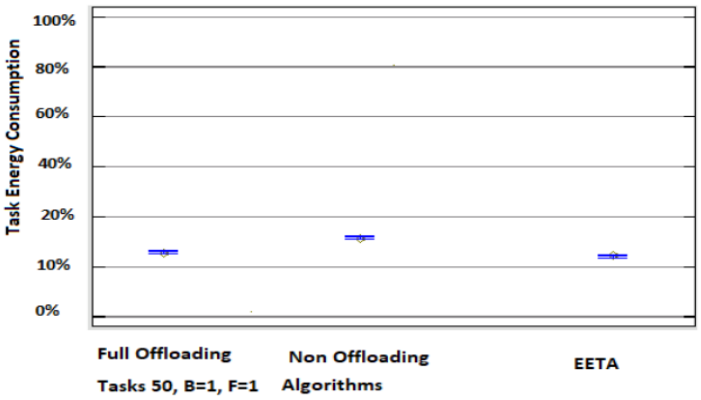

Figure2. Energy Consumption during the task offloading

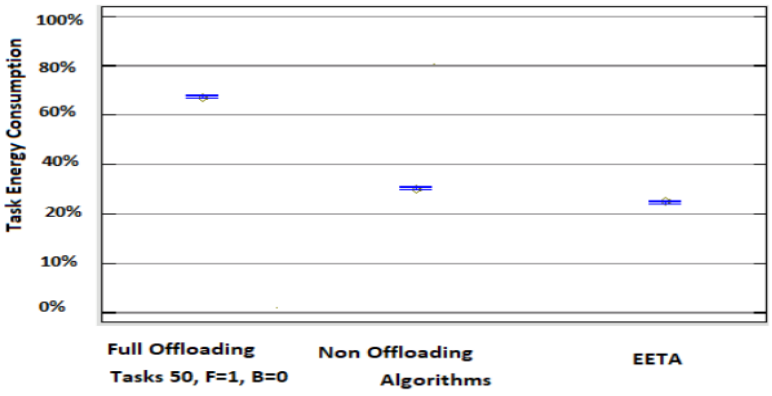

Figure3.Performance Analysis of Application Based on Tasks

However, service rate is represented and calculated via exponential distribution at the point of time. Figure 3 and 5 shows that after being tested the different application workload and granularity we have supposed to be reached to the final conclusion that every individual is required power consumption and it must be worked under given threshold value.

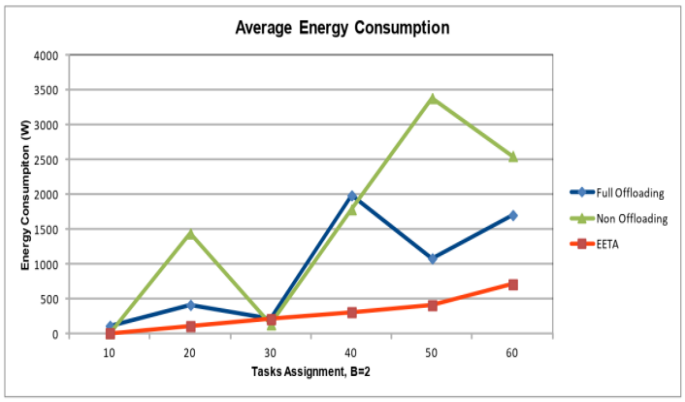

Figure4. Fixed Speed up factors comparison

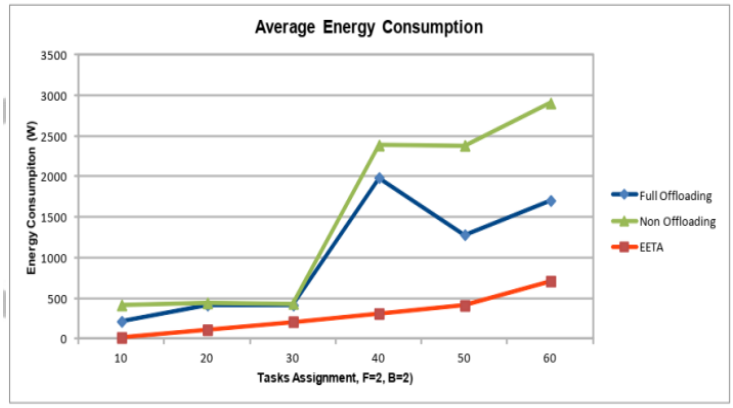

Figure5. Overall Energy Utilization

Whereas, proposed method EETA is again leading a better performance over traditional schemes.

Conclusion

The EETA framework proposed in this paper does not consider the power consumption on Idle and waiting time for task result and disk usage. Dynamic adoption could be happened during application execution life cycle I.e., network and device profiling change by over time. The preceding condition produces the inaccurate offloading result. In the future, we will propose dynamic and effective energy efficient algorithm which would be able to cope with environmental changes (e.g., bandwidth and workload change). They consider problem will be handled by dynamic optimization scheme.

Acknowledgment

The authors wish to thank A, B, C. This work was supported in part by a grant from XYZ.

References

1. Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N.: The case for VM-based cloudlets in mobile computing. IEEE Pervasive Computing 8(4), 14–23 (2013)
2. Dinh, H.T., Lee, C., Niyato, D., Wang, P.: A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Communications and Mobile Computing (2014)
3. Baliga, J., Ayre, R., Hinton, K.: Green cloud computing: Balancing energy in processing, storage, and transport. Proceedings of the IEEE 99(1), 149–167 (2011)
4. Barbera, M.V., Kosta, S., Mei, A., Stefa, J.: To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. In: Proc. of the IEEE INFOCOM (2013)
5. Yang, K., Ou, S., Chen, H.H.: On effective offloading services for resourceconstrained mobile devices running heavier mobile internet applications. IEEE Communications Magazine 46(1), 56–63 (2008)
6. Kumar, K., Lu, Y.H.: Cloud computing for mobile users: Can offloading computation save energy? Computer 43(4), 51–56 (2010)
7. Portokalidis, G., Homburg, P., Anagnostakis, K., Bos, H.: Paranoid Android: versatile protection for smartphones. In: Proc. of the 26th Annual Computer Security Applications Conference, pp. 347–356. ACM (2010)
8. Chen, E.Y., Itoh, M.: Virtual smartphone over ip. In: Proc. of the IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM), pp. 1–6. IEEE (2010)
9. Wen, Y., Zhang, W., Luo, H.: Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones. In: Proc. of the 2012 IEEE INFOCOM, pp. 2716–2720. IEEE (2012)
10. Rudenko, A., Reiher, P., Popek, G.J., Kuenning, G.H.: Saving portable computer battery power through remote process execution. ACM SIGMOBILE Mobile Computing and Communications Review 2(1), 19–26 (1998)
11. Tang, M., Cao, J.: A dynamic mechanism for handling mobile computing environmental changes. In: Proc. of the 1st International Conference on Scalable Information Systems, p. 7. ACM (2006)
12. Chun, B.G., Maniatis, P.: Dynamically partitioning applications between weak devices and clouds. In: Proc. of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, p. 7. ACM (2010)
13. Cuervo, E., Balasubramanian, A., Cho, D.K., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.: Maui: making smart phones last longer with code offload. In: Proc. of the 8th International Conference on Mobile Systems, Applications, and Services, pp. 49–62. ACM (2010)
14. .Yao ;Zhi sheng Niu Energy-efficient task offloading for multi user mobile cloud computing. (4), 15–23 (2013)
15. Energy-Efficient Mapping and Scheduling of Task interaction Graphs for Code offloading in Mobile Cloud Computing (2016)P. Balakrishnan; Chen-Khong Tham
16. Balakrishnan P, Tham CK. Energy-efficient mapping and scheduling of task interaction graphs for code offloading in mobile cloud computing. InProceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing 2013 Dec 9 (pp. 34-41). IEEE Computer Society.
17. Mobile cloud An Energy Efficient MCC Collaborative Framework With Extended Mobile Participation for Next Generation Networks John Panneerselvam ;James Hardy; Lu Liu; Bo Yuan;Nick Antonopoulos IEEE Access (2016)