Journal of Computer Science

Original Research Paper

An Optimized Mobile Cloud Computational Offloading Framework using K-Means Algorithm

[1]Meena Veeraiyan, [2]S. Kousika, [3]J. Senthilkumar and [1]Joy Christy Antonysami

[1]Deparment of Computer Science Engineering, SASTRA Deemed University, India
[2]Deparment of Information Technology, SASTRA Deemed to be University, India
[3]Department of Computer Science Engineering,
Srinivasa Ramanujam Centre Kumbakonam, SASTRA Deemed to be University, India

Article history
Received: 02-07-2019
Revised: 21-08-2019
Accepted: 30-01-2020

Corresponding author:
Joy Christy Antonysami
Department of Computer Science Engineering, SASTRA Deemed University, India
Email: joychristy001@gmail.com

Abstract: Offloading the execution of heavy computational modules from mobile devices to Mobile Cloud Computing (MCC) is inevitable in today's era as it mainly focuses in consuming less battery power and execution time. But, the problem incurred with identifying the most optimal cloud device to map each module still remains a challenge in cloud computing environment. In this paper, a novel MCC offloading framework is proposed to fasten the allocation and execution of high computational modules that runs in the mobile device, effectively on the cloud. The framework employs K-Means clustering algorithm to group the nearest cloud virtual machines that best suits for executing modules of software running in the mobile. The objective of the paper is to maximize the energy savings by extending the battery life and execution speed of mobile device when executing heavy computational modules. The optimal selection of cloud device is attained by grouping the requirements of each module with the nearest cloud devices offering the same requirements using K-Means Algorithm. The proposed framework is compared with the existing mobile computation offloading frameworks with respect to energy saving, execution time and energy consumption. The results show that the proposed work executes the modules of computationally intensive modules in minimum time span with maximized energy savings than the existing frameworks.

Keywords: Offloading, K-Means, Cloud Virtual Machines, Euclidean Distance

Introduction

Extensive use of smart phones encourages programmers to build plenty of software applications that cope up our day to day life. Some popular applications installed in smart phones include Car Remote Control, 2D Bar-Code Reader, Mobile Environmental Sensors, Mobile Security and Authentication, Cost Considerations, Medical Microscope and so on. These applications demand smart phone to act as a computing device as they evolve heavy computational modules (Kuang *et al*., 2018). But, the constraints with memory, processing power and battery life of smart phones degrades its performance down in terms of speed, battery power and efficiency. A technological solution to this issue is called offloading. The evolution of wireless communication and cloud computing allows smart phones to offload its high computational modules to be executed in a remote infrastructure through wireless communications.

Many researchers have proposed different kinds of offloading frameworks (Mazouzi *et al*., 2019; Zhang *et al*., 2019; Elgendy *et al*., 2019) While some offload the entire application, others select only heavy computational modules. The number of factors that needs to be considered for offloading the modules is network bandwidth, transmission cost, computational cost, available memory and latency. A decision engine in these frameworks randomly executes all the modules in both local and remote environment to decide whether to run in local device or in the cloud by taking into account of all the above factors. If a module is to be executed on cloud, it is more advantageous only when the remote execution saves energy without worsening the normal response time. The existing frameworks do not attempt to optimize the offloading with all these five factors. There are still number of aspects that energize global optimization in offloading task. This paper postulates yet another heavy computational module offloading framework through

machine learning approach. The novelty of the paper is to offload the heavy computational modules of the software on cloud virtual machines through descriptive analysis technique in data mining. The module that requires similar types of resources are grouped into one cluster and the node that is nearest to the module requirements has been selected for offloading. The results have proven that the proposed framework will efficiently identify the modules that should run on local and remote machines.

The remaining section of the paper is organized as follows: Section 2 presents the literature review, section 3 denotes the proposed methodology (MCCO framework), section 4 depicts the experimentation and result analysis and finally section 5 discusses the findings and conclusion of the paper.

Review of Literature

Computational offloading has serious attention by industrialist and academician to find optimal offloading in mobile cloud computing in order to improve the performance of resource constraint devices. The existing works used optimization algorithm has provide the solutions for optimal mapping between intensive jobs and executing servers. Achary *et al*. (2015) proposed dynamic job scheduling to offload the intensive modules in cloud server using ant colony optimization in order to improve throughput and quality of services and also addressed the problem when mobile is used in remote server and Hadoop environment. Ramezani *et al*. (2015) proposed Multi-Objective Particle Swarm Optimization (MOPSO) and Multi-Objective Genetic Algorithm (MOGA) to find optimal offloading in cloud in order to decrease job response time, makespan and the cost of the service providers. Guo *et al*. (2012) proposed a heuristic algorithm called particle swarm optimization for task scheduling in cloud environments with minimum cost of processing with less focus on energy saving and service level agreement.

Table 1: Denotes the review of similar works along with its advantages and disadvantages in brief

No.	Title	Authors	Technique used	Datasets	Metrics	Advantages	Drawbacks
1	Computation Offloading in Hand-Held Devices Using Ternary Decision Maker in Accountance with Time and Energy	Chourasiya and Singh (2018)	Ternary Decision Marker	Standard Android Cloud	Time, Energy	✓ Reducing computation power in mobile cloud ✓ Providing security to offloaded data	Not concentrate on Heterogeneous offloading
2	Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading	You and Huang (2017)	Multiuser Mobile Edge Computation Offloading	Dedicated Dataset	Energy	✓ Considered both finite, infinite clod capacity for resource allocation ✓ Depends on energy consumption and channel gains deriving offloading priority function	Diminished Performance due to higher power saving
3	A fast hybrid multi-site computation offloading for mobile cloud computing	Goudarzi *et al*. (2017)	Optimized multisite PSO	DB, RayTrace, JESS, R4	Cost, energy, Time	✓ Reduces search space for optimal offloading solution	Energy consumption is more not suitable for battery operated devices
4	A Heuristic Algorithm for Multi-site Computation Offloading in Mobile Cloud Computing Hill Climbing	Enzai and Tang (2016)	Synthetic data	Time, energy, Cost		✓ Minimizes energy consumption ✓ Minimizes computation time ✓ Minimizes total computation cost	Scalability
5	A hybrid heuristic queue based algorithm for task assignment in mobile cloud	Rashidi and Sharifian (2017)	Ant-Colony genetic	Video Encoding	Time, Energy	Decreasing mean completion time, total energy consumption Decreasing number of dropped tasks	-
6	Modeling multi-factor multi-site risk-based offloading for mobile cloud computing	Wu and Huang (2014)	Multifactor-Multisite Risk based offloading	Dedicated Dataset	Time, Energy	✓ Aggregate overall offloading benefits and risks	✓ Diminished Performance ✓ Non-optimized resource handling
7	An energy-efficient multisite offloading algorithm for mobile devices	Niu *et al*. (2013)	Multi-way graph partitioning	Established the Random Graphs	Time, Energy	✓ Minimizing energy consumption and execution time ✓ Better adaptability to wireless networks	-
8	Task scheduling optimization in cloud computing based on heuristic algorithm	Guo *et al*. (2012)	Particle swarm optimization	Established random tasks	Time	✓ Minimize cost of processing	Not addressing Energy saving and service level agreement.
9	Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments	Ramezani *et al*. (2015)	Multi-Objective Particle Swarm Optimization (MOPSO) and Multi-Objective Genetic Algorithm (MOGA)		Service time, service cost, QOS	✓ Reduces job response time and makespan ✓ Decrease cost to providers	-
10	Dynamic job scheduling using ant colony optimization for mobile cloud computing	Achary *et al*. (2015)	Ant Colony optimization	Random generation of wireless nodes	Throughput, QOS	✓ Dynamically scheduling the tasks ✓ Improve throughput and quality of service	

Merits and demerits of the existing mobile offloading frameworks is shown in Table 1.

Many authors focused on executing intensive jobs on multiple cloud servers. Niu *et al*. (2013) proposed a novel scheme multi-way graph partitioning to have energy efficient multi-site offloading in MCC by generating random graphs that provides solution with minimized energy consumption. The authors claimed that the proposed work is more adaptable to wireless networks. Wu and Huang (2014) proposed a multifactor multisite risk based offloading in MCC to select suitable nodes for offloading by analyzing the overall benefits and risks of the system. Rashidi and Sharifian (2017) proposed a novel scheme of hybrid heuristic queue based algorithm for task assignment by implementing ant colony and genetic optimization algorithms with decreased mean completion time, total energy consumption by evaluation the factors such as time and energy incurred in the video encoding system.

Enzai and Tang (2016) implemented hill climbing algorithm to offload computational task in multisite cloud servers with decreased energy consumption, computation time and total computational cost without considering the scalability factor. Mohammad Goudarzi *et al*. (2017) proposed fast hybrid multisite computational offloading (FHMCO) by implementing particle swarm optimization algorithm to obtain near optimal offloading solution with decreased cost, energy and time factors. They proved their solutions on applications like JESS, DB and RayTrace. You and Huang (2016) proposed energy efficient resource allocation scheme for mobile edge computational offloading with the considerations of finite and infinite capacity of cloud resources. Here channel is gained by using priority function. Chourasiya and Singh (2018) proposed novel scheme Ternary Decision Maker (TDM) to select appropriate target device either phone or cloud for offloading process in order to obtain reduced computing power in mobile devices.

Proposed Methodology

Graph Construction

Each heavy computation application is viewed as weighted directed graph G = (Ve, Ed) in which vertices Ve contains computation cost of each module in application and edges Ed contains transmission data cost among the application units. In each vertex has set of weights (w_I, m_i) in which w_i represents computation cost of the component '*i*' on the mobile device, m I represents memory consumption of the component '*i*'. Figure 1 depicts the sample weighted graph for multi cluster offloading problem.

Each cluster is configured with their CPU capacity, memory usage, memory available, Speedup factor etc. Group of clusters are represented as C = (c1, c2,…..cm). In each cluster is represented as (CPUi, MCi). Figure 2 depicts an example of group of clusters with their configuration.

Multi Cluster Computation Offloading Problem Formulation

In this section we formulate multi cluster offloading problem based on computation time requirement of individual module and memory requirement of individual module.

Execution Cost Model

The execution time application model aims at finding the best clustering of an application graph Z satisfying Y = min(TE(Z)). Execution time for i^{th} module is given in following formula:

$$E(i) = T(i) + \left(Num\_of\_inst(i) / Sf(j)\right) \quad (1)$$

where, $T(i)$ denote the transmission cost of *i*th module and $Sf(j)$ denote the *j*th cloud capacity.

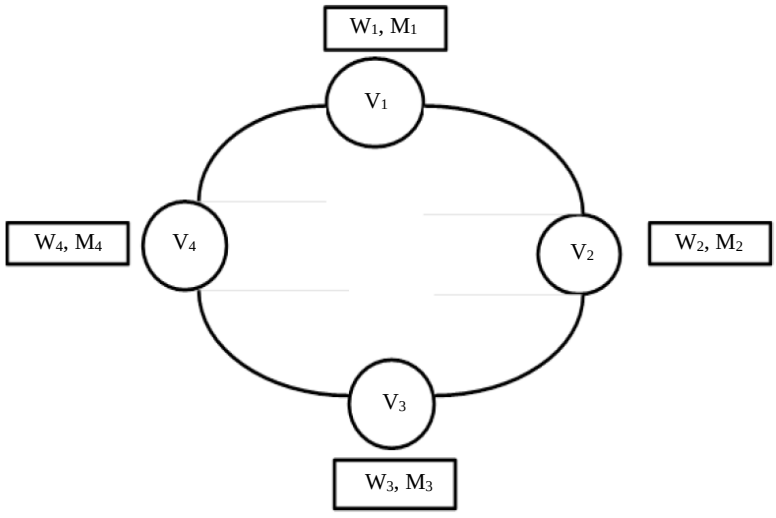

Fig. 1: Weighted directed graph of heavy computation application

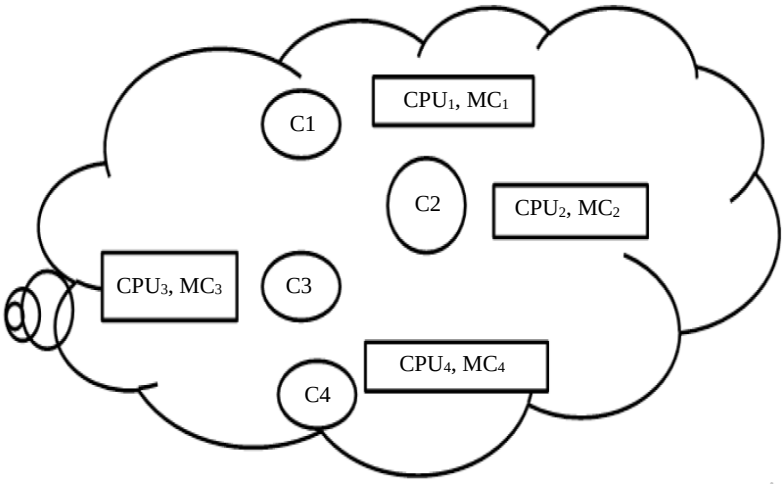

Fig. 2: Cluster configuration for effective offloading

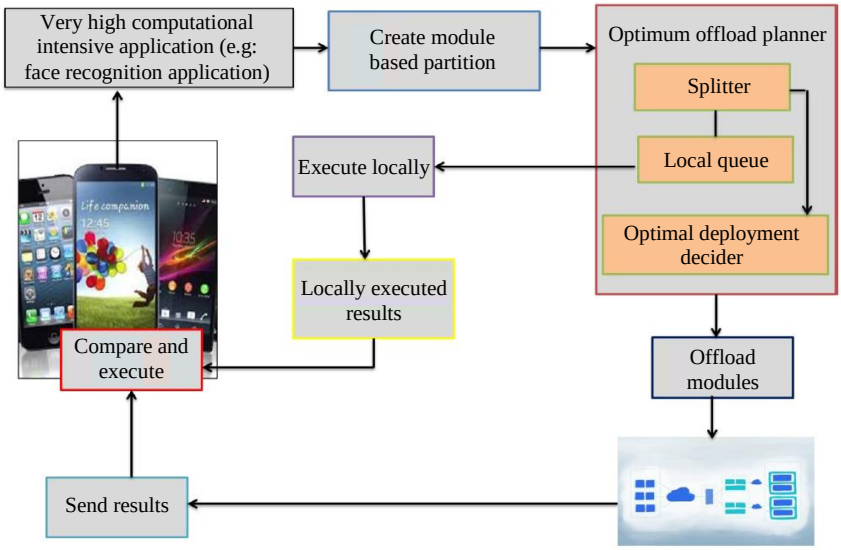

Fig. 3: MCCO architecture

Multi Cluster Computation Offloading Architecture

In this section we propose a Multi cluster Computation Offloading Solution called MCCO, Which finds appropriate cluster to deploy the offloaded module in timely manner. The proposed solution includes two methods based on computation time and memory of the individual module. In order to achieve optimal deployment, we proposed Input: Available resources in CVM Output: Clusters that maps user requirements with cloud resources Pre: Computational cost and storage requirements of each module in the application 1. Set k to number of modules 2. Set module requirements as initial centroids 3. Do Compute distance between initial centroids and CVM resources Group the nearest cloud resources into the cluster with minimum distance Recompute centroids 4. Until there is no change in the cluster group or cluster centroids.

MCCO Architecture

MCCO takes the partitioned modules of high computational software from mobile device and passes them to an optimum offload planner (OOP). OOP process the modules under three stages splitter, local queue and optimal deployment Decider. Figure 3 denotes the components of the proposed framework and their relationships.

The splitter is responsible for partitioning the applications as two major sections namely local queue and optimal deployment decider, where, the first section contains a list of modules that are executed in mobile device. The next section contains a list of heavy computational modules for computation offloading. The local queue sends the locally executable modules to mobile for their execution. The optimal deployment planner of the proposed work employs K-Means

clustering algorithm (Meilă, 2019; Heil *et al*., 2019; Gan and Ng, 2017) for the efficient allocation of modules over multiple Cloud Virtual Machines (CVMs). The algorithm maps individual module requirements with available cloud resources. K-Means takes the requirements of individual module as initial cluster centroids, such as computational time and storage capacity and tries to find the appropriate cloud virtual machine to execute them. The distance between the module requirements and the available services are computed using Euclidean distance. Finally, the CVMs that are offering similar types of services in par with the requirements of a module are grouped into one cluster. The modules are then assigned into the nearest cloud virtual machine for execution. Algorithm 1 denotes the clustering process of the proposed work.

Algorithm 1: *Cloud Service Selection*

Input: Available resources in CVM
Output: Clusters that maps user requirements with cloud resources
Pre: computational cost and storage requirements of each module in the application

1. Set k to number of modules
2. Set module requirements as initial centroids
3. Do
   Compute distance between initial centroids and CVM resources
   Group the nearest cloud resources into the cluster with minimum distance
   Recompute centroids
4. Until there is no change in the cluster group or cluster centroids

In this work, optimization of offloading task is highly concerned with reduced transmission cost. Thus, a module with reduced number of instructions that occupies less memory space is sent as a relay to each virtual machine in the cloud. The time taken to transmit the relay module is used to compute the time taken for transmitting all modules to the available cloud virtual machines. Among the several cloud virtual machines in the cluster, a device with reduced transmission cost is chosen for running that module. Algorithm 2 is proposed to select the best virtual machines by considering the transmission cost discussed in Algorithm 3.

Algorithm 2: vmSelect

```
Input:  Modules with their computation time, memory
        requirement Cloud service with cpu capacity
Output: Optimal mapping of modules with suitable
        virtual machine in suitable cloud server.
1.      res = 0;vm[1,…no of modules];map[1,…..no of
        modules];
2.      min=inf;
3.      while i=1 to number of modules
4.          while j=1 number of cluster
5.              while k= number of vm's in j
6.                  calculate E[i] using eq 1
7.                  if(min<E[i])
8.                                  map[i]=j
9.                                  vm[i]=k
10.                                 min=E[i]
11.                 end if
12.             end for
13.         end for
14.         E[i]=min
15.     end for
```

Algorithm 3: tcCost

Input: Total number of modules (m1,m2,m3…..mn) in M; Total number of Virtual machines in suitable cloud server, VM; Computation time required for each module E(i);
Output: calculating transmission cost for one module on every virtual machines in cloud server.

```
1.  while j=1 to VM
2.      E[1][j]=m1 is executed in vmj
3.      tr[1][j]=transmission cost is calculated
    end
4.  while i=2 to M
5.      for j=1 to VM
6.          diff=size[i]/size[1];
            tr[i][j]=tr[1][j]*diff;
            E[i][j]=E[1][j]*diff;
11.     end
12. end
```

Experimentation and Results Discussions

The experimentation is conducted over face recognition software with 20 modules each of which is different from size, number of instructions, CPU and memory usages. Table 2 depicts name, computational and memory usage of each module in the experimental software. The software has been repeatedly executed on Ubuntu 14.04LTS and the average values of each parameter have been taken for multi-site offloading. The CPU clock frequency of the cloud devices is set from 1.5 GHz to 2 GHz. The bandwidth between cloud virtual machines and mobile device is ranging from 200 Kbps to 1000 kbps and the total bandwidth varies from 5Mbps to 20 Mbps.

The section demonstrates the advantages of the proposed MCCO framework in mobile cloud computing offloading scheme by comparing the execution time and energy consumption with two most popular existing approaches namely Dynamic Programming Mobile Offloading Framework and Greedy Mobile Offloading Framework algorithms.

From the twenty modules of experimental software, a module that requires minimum storage with reduced number of instructions has been chosen as a relay module and transmitted over the CVMs to examine the execution time and transmission cost. The outputs are then used to compute the execution time and transmission for the remaining 19 modules. Having the requirements of the modules as cluster centroids, the information gained from CVMs are clustered to find the nearest optimal VM to execute the module. The modules are mapped to the nearest cloud agent so as to ensure the minimum execution time and maximum energy saving.

Energy Saving

The modules are sent through different network bandwidths ranging from 5000 Kbps to 15000 Kbps for analyzing the energy saving capacity of MCCO framework. Figure 4 shows the performance of the proposed work with respect to energy consumption. From the results it has been observed that the proposed MCCO framework consumes less energy when the bandwidth increases (i.e.,) more the bandwidth increases more energy is saved. In addition, when the total number of bandwidth B is assigned, there are more suitable devices found in CVM clusters that meet the requirements of the module.

Table 2: Description of experimental software

S. no	Module (function) name	Computational time(ns)	Memory (KB)
1	Main	117535194	2300
2	Add Widgets	44254512	3190
3	FaceRwcView	175646814	3543
4	SimpleController	26371082	3750
5	propertyChange	21943	6015
6	getIcon	132694629	6311
7	ActionPerformed	58124143844	188148
8	SetImage	26947	124918
9	ValidateTextField	105480	124918
10	ValidateFileSelection	3307213	124918
11	ValidateFolderSelection	103555	124918
12	SimpleValidator	3651368	124918
13	ImageDistanceInfo	1540	157879
14	CheckImageSizeCompatibility	3448494	153947
15	getImageData	5898010	153947
16	Matrix2D	3711808	157879
17	getDistance	50815	157879
18	MatchResult	19404032	41905087
19	HandleUserInputs	428910526	157898
20	SimpleButtonListener	430443061	157898

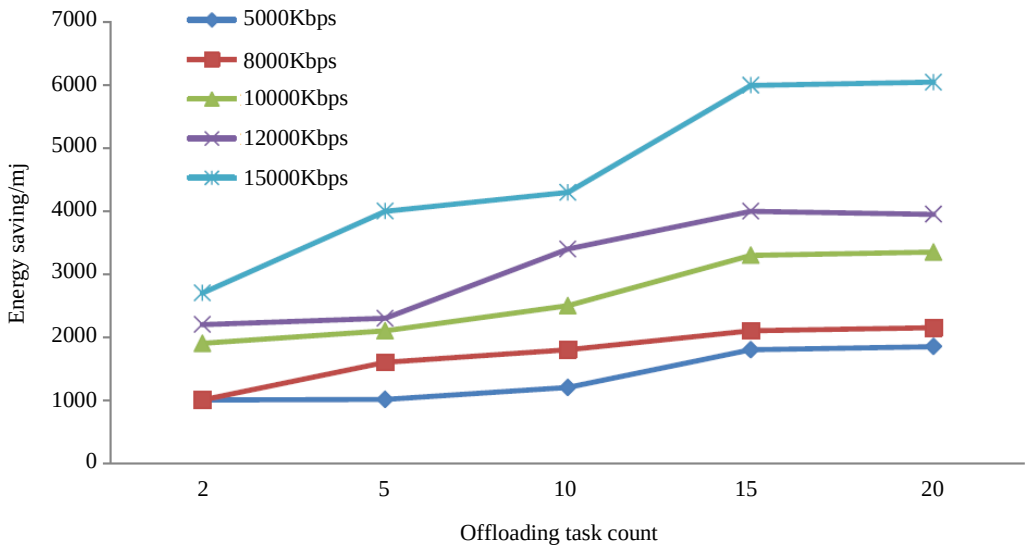

Fig. 4: MCCO Energy saving analysis

Execution Time

Figure 5 denotes the execution time analysis of MCCO framework with varying bandwidths. The results explicit an inverse correlation in the output as the execution time of the proposed algorithm is minimum when the module is passed through higher bandwidths. However, according to the size and bandwidth, the execution time of the MCCO may vary. For instance, the smallest module offloading tasks given in the scenario is 50, which is sent through 20000Kbps is comparatively low than the other bandwidths in terms of execution time. The pace of the execution time is increasing gradually even with higher bandwidths and the persistent level of execution time is hardly achieved.

Figure 6 shows the analysis on the energy consumption over the execution of face recognition software in remote and MCCO frameworks. From the results, it is observed that the local execution of all modules consumes more energy as the device has limited capacities in terms of processor and battery life. When, all modules are offloaded onto remote machines, the energy saving is quite high than executing all modules in local machine which is unrealistic. The MCCO framework based execution of software has shown a significant improvement in energy saving as it runs the modules that require minimum resources in the local devices and energy consuming modules in the remote machines, thus saving the transmission cost.

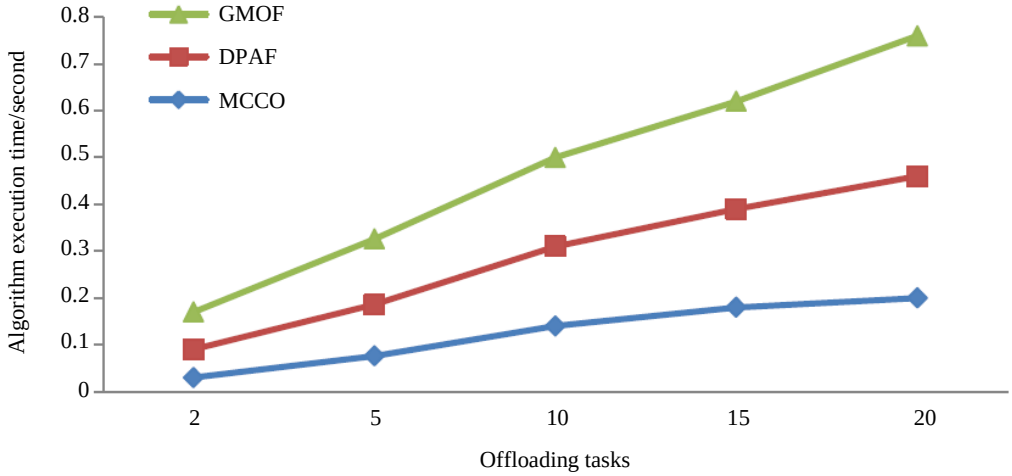

Fig. 5: MCCO execution time analysis

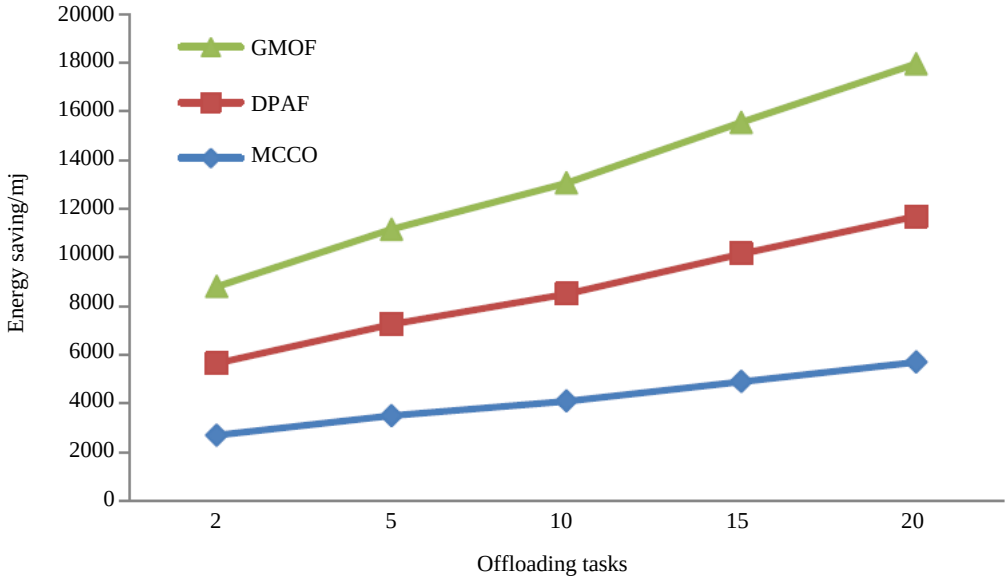

Fig. 6: MCCO energy consumption analysis on remote machines

Conclusion

The paper proposes a cloud virtual machines based computation offloading framework for outsourcing the modules of software that runs in the mobile platform to cloud environment. The framework considers minimum completion time as a major constraint to ensure offloading realistic. The framework also employs K-Means clustering algorithm to identify suitable VMs in cloud environment based on the requirements of each module such as time and storage. The optimized allocation is then achieved by selecting a VM with minimum transmission cost from the cluster. Simulation results discloses that the proposed offloading framework outperforms the traditional LDR and DPAF offloading optimization algorithms and also with all local execution schemes in terms of fast execution and energy savings. In future, this work can be extended to explore complicated offloading of modules from multiple software in real time environment.

Acknowledgment

The authors would like to thank SASTRA Deemed to be University for the facilities used in this work.

Author's Contributions

All authors equally contributed to the final version of the manuscript.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and there are no ethical issues involved.

References

Achary, R., V. Vityanathan, P. Raj and S. Nagarajan, 2015. Dynamic Job Scheduling using Ant Colony Optimization for Mobile Cloud Computing. In: Intelligent Distributed Computing, Buyya, R. and S. Thampi (Eds.), Springer, Cham, pp: 71-82.

Chourasiya, N.L. and T.P. Singh, 2018. Computation offloading in hand-held devices using ternary decision maker in accountance with time and energy. Proceedings of International Conference on Recent Advancement on Computer and Communication, (ACC' 18), Springer, Singapore, pp: 595-607. DOI: 10.1007/978-981-10-8198-9_62

Elgendy, I.A., W. Zhang, Y.C. Tian and K. Li, 2019. Resource allocation and computation offloading with data security for mobile edge computing. Future Generat. Comput. Syst., 100: 531-541. DOI: 10.1016/j.future.2019.05.037

Enzai, N.I.M. and M. Tang, 2016. A heuristic algorithm for multi-site computation offloading in mobile cloud computing. Proc. Comput. Sci., 80: 1232-1241. DOI: 10.1016/j.procs.2016.05.490

Gan, G. and M.K.P. Ng, 2017. K-means clustering with outlier removal. Patt. Recog. Lett., 90: 8-14. DOI: 10.1016/j.patrec.2017.03.008

Goudarzi, M., M. Zamani and A.T. Haghighat, 2017. A fast hybrid multi-site computation offloading for mobile cloud computing. J. Netw. Comput. Applic., 80: 219-231. DOI: 10.1016/j.jnca.2016.12.031

Guo, L., S. Zhao, S. Shen and C. Jiang, 2012. Task scheduling optimization in cloud computing based on heuristic algorithm. J. Netw., 7: 547-547. DOI: 10.4304/jnw.7.3.547-553

Heil, J., V. Häring, B. Marschner and B. Stumpe, 2019. Advantages of fuzzy k-means over k-means clustering in the classification of diffuse reflectance soil spectra: A case study with West African soils. Geoderma, 337: 11-21. DOI: 10.1016/j.geoderma.2018.09.004

Kuang, Z., S. Guo, J. Liu and Y. Yang, 2018. A quick-response framework for multi-user computation offloading in mobile cloud computing. Future Generat. Comput. Syst., 81: 166-176. DOI: 10.1016/j.future.2017.10.034

Mazouzi, H., K. Boussetta and N. Achir, 2019. Maximizing mobiles energy saving through tasks optimal offloading placement in two-tier cloud: A theoretical and an experimental study. Comput. Commun., 144: 132-148. DOI: 10.1016/j.comcom.2019.05.017

Meilă, M., 2019. Good (K-means) clusterings are unique (up to small perturbations). J. Multivariate Anal., 173: 1-17. DOI: 10.1016/j.jmva.2018.12.008

Niu, R., W. Song and Y. Liu, 2013. An energy-efficient multisite offloading algorithm for mobile devices. Int. J. Distributed Sensor Netw., 9: 518-518. DOI: 10.1155/2013/518518

Ramezani, F., J. Lu, J. Taheri and F.K. Hussain, 2015. Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments. World Wide Web, 18: 1737-1757. DOI: 10.1007/s11280-015-0335-3

Rashidi, S. and S. Sharifian, 2017. A hybrid heuristic queue based algorithm for task assignment in mobile cloud. Future Generat. Comput. Syst., 68: 331-345. DOI: 10.1016/j.future.2016.10.014

Wu, H. and D. Huang, 2014. Modeling multi-factor multi-site risk-based offloading for mobile cloud computing. Proceedings of the 10th International Conference on Network and Service Management (CNSM) and Workshop, Nov. 17-21, IEEE Xplore Press, Rio de Janeiro, Brazil, pp: 230-235. DOI: 10.1109/CNSM.2014.7014164

You, C. and K. Huang, 2016. Multiuser resource allocation for mobile-edge computation offloading. Proceedings of the IEEE Global Communications Conference, Dec. 4-8, IEEE Xplore Press, Washington, DC, USA, pp: 1-6. DOI: 10.1109/GLOCOM.2016.7842016

Zhang, F., J. Ge, C. Wong, C. Li and X. Chen *et al*., 2019. Online learning offloading framework for heterogeneous mobile edge computing system. J. Parallel Distributed Comput., 128: 167-183. DOI: 10.1016/j.jpdc.2019.02.003