Dynamic Resource Allocation in Hybrid Mobile Cloud Computing for Data-Intensive Applications

Mohammad Alkhalaileh, Rodrigo N. Calheiros, Quang Vinh Nguyen, and Bahman Javadi

School of Computing, Engineering and Mathematics, Western Sydney University, Sydney, Australia
{mohammad.nour, r.calheiros, q.nguyen, b.javadi}@westernsydney.edu.au

Abstract. Mobile cloud computing is a platform that has been used to overcome the challenges of mobile computing. However, emerging data-intensive applications, such as face recognition and natural language processing, imposes more challenges on mobile cloud computing platforms because of high bandwidth cost and data location issues. To overcome these challenges, this paper proposes a dynamic resource allocation model to schedule data-intensive applications on integrated computation resource environment composed of mobile devices, cloudlets and public cloud which we refer as hybrid mobile cloud computing (hybrid-MCC). The allocation process is based on a system model taking into account different parameters related to the application structure, data size and network configuration. We conducted real experiments on the implemented system to evaluate the performance of the proposed technique. Results demonstrate the ability of the proposed technique to generate an adaptive resource allocation in response to the variation on application data size and network bandwidth. Results reveal that the proposed technique improves the execution time for data-intensive applications by an average of 78% and saves the mobile energy consumption by an average of 87% in compared to using only a mobile device while the monetary cost increased only 11% due to using cloud resources and mobile communication.

Keywords: Hybrid Mobile Cloud Computing · Data-Intensive Mobile Applications · Offloading Technique · Mobile Application Scheduling · Resource Allocation · Application Execution Modelling

1 Introduction

The use of mobile devices such as smartphones and tablets underwent a tremendous increase due to the advancement in functionalities supported by enhanced features such as high connectivity, faster CPU, large memory, and sophisticated sensors. In 2019, it is estimated that 46% of the internet traffic will be generated by mobile devices with an estimated 30.6 Exabyte generated monthly [7].

Although mobile devices are now equipped with considerable high-performance computation resources, they still face major challenges in meeting the requirements of computation-intensive and data-intensive mobile applications.

Cloud Computing has been extensively proposed to overcome the shortcomings of mobile computing. Cloud computing provides computation and storage as services in a highly scalable and secure manner. Satyanarayanan et al. [19] argue that cloud computing model is potentially the best solution to solve the deficiencies in mobile device resources. The integration of mobile computing and cloud computing is known as Mobile Cloud Computing (MCC) [5]. MCC aims to augment mobile devices by improving and optimizing their computing capabilities while performing compute-intensive tasks in cloud-based resources [22]. Migrating resource-intensive computations from smartphone devices to the cloud via wireless communication technologies are refer to the concept of computation offloading. Computation offloading has been studied intensively in the literature and different techniques have been proposed for optimizing energy consumption and meeting user Quality of Service (QoS) metrics such as response time and monetary cost. These techniques include computation augmentation [1], device cloning [6], nearby-resource computation [19], provisioning middleware [10,12,17] and context-aware and profiling [13,22]. While there are some works about computation offloading, most of them are limited to computation-intensive applications which confined on simple MCC environment.

In this work, we propose a dynamic resource allocation model to schedule data-intensive applications on integrated computation resource environment composed of mobile devices, cloudlets and public cloud which we refer as hybrid mobile cloud computing (hybrid-MCC). We consider hybrid-MCC to construct new efficient MCC resource models. Moreover, we model a ***data-intensive*** mobile application scheduling and allocation as a multi-objective optimization problem for hybrid-MCC. Data-intensive mobile applications could be face recognition, data analytics and natural language processing [2]. The work contribution can be listed as:

- Multi-objective optimization of device energy and monetary cost on the hybrid-MCC environment under the constraints of mobile device energy and task deadline.
- Propose a data-aware offloading technique for data-intensive applications on hybrid-MCC environment.
- Performance evaluation of the proposed technique using real experiment and simulation under various working conditions.

The rest of the paper is structured as follows. In Section 2, we investigate the related work on task scheduling and allocation optimization on MCC. Section 3 presents an overview of the system architecture as a hybrid-MCC environment. Application execution models and problem formulation are described in Section 4, while the proposed data-aware offloading technique explained in Section 5. The model performance evaluation and experimental results are discussed in Section 6. Section 7 provides conclusions and future work.

2 Related Work

In this section, we review the existing works on task scheduling and allocation optimization on MCC from different perspectives based on different optimization objectives.

It is critical for mobile applications to deliver complex high-performance functionalities at a lower energy level. Mobile device I/O processing and network communications are energy-hungry components [1]. Offloading heavy tasks to the cloud can reduce energy consumption in an efficient way. Abolfazlia et al. [1] showed that mobile computation augmentation can deliver intensive computation to mobile users, save device energy use and prolong battery life. Different energy-based models have been proposed in literature such as: mobile device cloning in remote resources [6], code offloading and migration [8], application and network profiling [11], and application decomposition and reusability [10, 15].

However, the aforementioned energy-based MCC solutions focus on application code complexity migration for device energy optimization and do not consider the variation of context parameters such as input data size, network bandwidth, and corresponding data communication cost. Mobile network performance has a significant contribution to improving application responsiveness and thus optimizing the energy consumption [4]. Offloading to nearby resources such as cloudlets [19] can enhance application responsiveness and availability. The decision to run an application locally or remotely is complicated and requires steady monitoring of network conditions and application profiling [10].

Data-intensive applications such as customized data analytics services, natural language processing, and face recognition are resource-intensive applications that require machines with powerful CPUs and huge memory space to load dynamic application code and data. These applications bring additional challenges for energy and cost optimization [20]. Processing large data files on mobile devices has direct overhead on device energy, whereas transferring large data files over mobile networks can increase the device idle time, as well the total computation cost and time. The literature shows high attention to resource elasticity and scalability. Resource-based MCC optimization models focus on cloud resource scaling and VM parallelism [12], elastic applications segmentation and deployment on hybrid-MCC and workflow-aware execution partitioning [17]. Sanaei et al. [18] highlighted the significance of having intelligent and scalable context-aware systems for mobile cloud applications that are capable of handling dynamic mobile environment.

For cost optimization, Nan et al. [16] studied the challenges of data-intensive mobile applications. The study uses monetary cost optimization as a significant factor to enhance user QoS. Minimizing monetary cost and enhancing customers QoS require an efficient cost optimization model. Zhou et al. [21] proposed a three-tier MCC middleware that empowers programmers with computation alternatives based on the application cost model and the offloading decision maker. Even though the proposed model includes the task data size in generating an optimized execution application tasks plan, the data size is marginally small and cannot reflect the scenario of data-intensive application scheduling on MCC.

The main common issue among the researches above is the lack of consideration the contribution of data size and MCC application complexity in building MCC architectures. To overcome this shortcoming, we propose an optimization technique for execution data-intensive MCC applications on the hybrid-MCC environment.

3 System Architecture

In this paper we propose a data-intensive mobile application offloading optimization framework on hybrid-MCC environment. The environment leverages three types of resources, namely, public cloud, cloudlet and mobile device. The public cloud provides high powerful and scalable resource. Cloudlet provides computation service to the mobile clients for sensitive requests. These resources can be accessed via Wi-Fi or cellular network. For a mobile device, we assume its ability to perform local computation and storage under the constraints of energy and wireless interface.

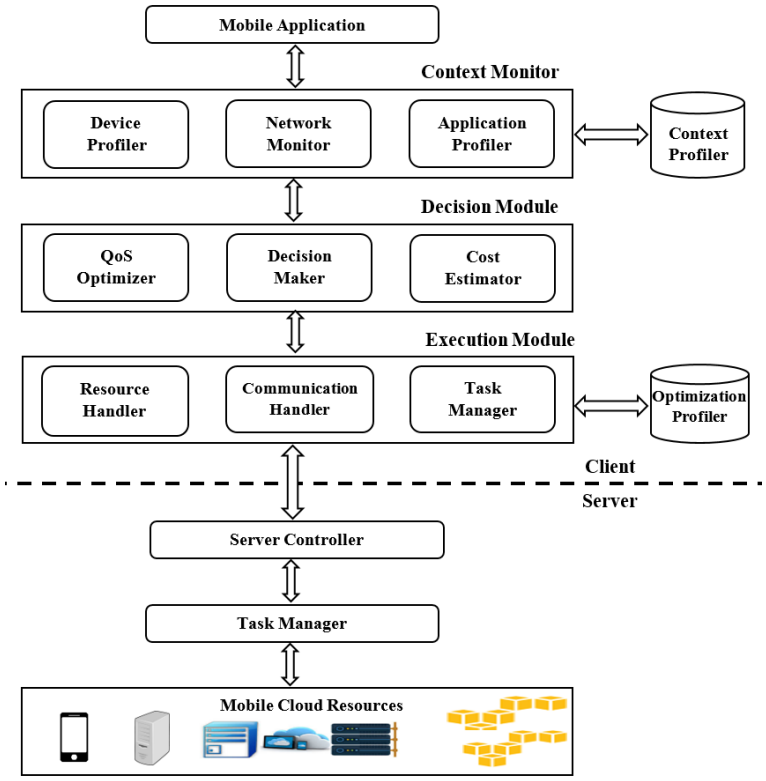


Fig. 1. Proposed hybrid-MCC offloading decision optimization framework.

The proposed offloading optimization framework consists of components that provides services of context monitoring, decision making and application execution. Fig. 1 illustrates the proposed framework.

The context monitor module is responsible for profiling context parameters at run time and supports the decision maker with the energy and monetary cost estimations. The framework offers three profilers, namely, device profiler for energy usage, network monitor for available mobile network information and application profiler to record the heuristic data about application execution with awareness to context information of network bandwidth.

The decision-making module is a QoS optimizer which aims to find the best application execution plan on a solution space where each solution is estimated by the cost estimator. The decision is based on user QoS and estimated execution time, total device energy and monetary cost.

The execution module is responsible for running the received application execution plan from the decision maker. It has three main components: The resource handler to manage access to resources for computation and storage, the communication handler to manage the communication networks between resources and task manager for running application tasks.

4 System Modelling and Problem Formulation

This section provides the modelling of data-intensive applications and the hybrid-MCC environment. Table 1 describes the mathematical notations used in the problem formulation.

4.1 Task Modelling

We assume the representation of a data-intensive MCC application A as set of independent tasks. An application A is modelled as:

$$A = \{t_1, t_2, \ldots, t_n\} \tag{1}$$

where n is number of tasks. Each task t_i is modelled as:

$$t_i = \{L_i, s_i, w_i, \partial_i\} \tag{2}$$

We have included data size and location parameters in task modelling to serve the objective of building a data-aware optimization model for scheduling mobile application tasks on a hybrid-MCC environment.

4.2 Resource Modelling

The system assumes three computation resources, namely, public cloud, cloudlets and mobile devices. A mobile device P_m is modelled as:

$$P_m = \{\alpha, \beta_{down}, \beta_{up}, \beta_{cost}, d, e, w_m\} \tag{3}$$

Table 1. Problem formulation notations

Symbol	Definition
t_i	Application Task i
L_i	Task input file location, either locally or remotely
s_i	Task input size
w_i	The number of task execution instructions
∂_i	Task deadline
β_{down}, β_{up}	Available network download and upload bandwidth
β_{cost}	The monetary cost of data communication using mobile network
d	Mobile device storage
e	Available device energy (Joule)
α	Available device memory
w_m	The device processing power
p_{cost}	Cost of processing in a cloud machine
w_{cloud}	Processing power of a cloud machine
C	Estimated total monetary cost of the application
E	Estimated total energy consumption in the mobile device
D_{t_i}	Estimated execution time for task t_i
ω_i	Task t_i data size sensitivity factor
β_s, β_r	Network bandwidth between data location and computation target
l	The network latency
$D_{t_i}^W$	The task waiting time in a remote server
λ	The mean rate of arrival task execution request to a remote server
L_q	The mean number of task requests in the queue
M^W	Application waiting time

A mobile device is connected to a cloudlet and the public cloud via Wi-Fi or cellular networks. A cloudlet or public cloud virtual machine P_{cloud} is modelled as:

$$P_{cloud} = \{\beta_{down}, \beta_{cost}, p_{cost}, w_{cloud}\} \tag{4}$$

4.3 Application Execution Models

This section describes the cost estimation models involved in formulating the mobile application scheduled to run the application tasks in the hybrid-MCC environment. In order to find the application execution plan, three estimation values need to be calculated, namely, task execution time, total mobile energy consumption and total monetary cost.

Task Execution Time Model The task execution time for task t_i is the sum of task processing time $D_{t_i}^P$ in the target computation environment P_{target}, data communication time $D_{t_i}^C$ and task average waiting time D^W for remote execution which can be calculated by using Little's rule [14].

$$D_{t_i} = D_{t_i}^P + D_{t_i}^C + D^W \tag{5}$$

$$D_{t_i}^P = w_{i,target} + (s_i * \omega_i) \tag{6}$$

$$D_{t_i}^C = \frac{s_i}{min(\beta_s, \beta_r)} + l \tag{7}$$

$$D^W = \frac{L_q}{\lambda} \tag{8}$$

Mobile Device Energy Model The energy consumption in the mobile device E is estimated by calculating E^P the total processing energy consumed by the mobile device, E^W the waiting energy particularly when the local execution time is less than the remote execution time, since the system assumes parallel tasks execution among the computation environment and E^C which is mobile energy consumption for data transfer communication.

$$E = E^C + E^P + E^W \tag{9}$$

$${E_i}^P = {D_{t_i}}^P * {\epsilon_i}^P \tag{10}$$

$$E^P = \sum_{i=1}^{m} {E_i}^P \tag{11}$$

$$M^W = max(0, (\sum_{i=1}^{m} {D_{ti}}^P - max(\sum_{i=1}^{c} {D_{ti}}^P, \sum_{i=1}^{cl}))){D_{ti}}^P)) \tag{12}$$

$$E^W = M^W * \epsilon^W \tag{13}$$

$$E^C = \sum_{i=1}^{m} {D_{ti}}^C * \epsilon^C \tag{14}$$

Where:

- $\sum_{i=1}^{m} {D_{ti}}^P$, $\sum_{i=1}^{c} {D_{ti}}^P$, $\sum_{i=1}^{cl} {D_{ti}}^P$: are the total processing time for all tasks executed locally (mobile device) and remotely (the public cloud and cloudlet), respectively. We consider the waiting energy consumption only if the waiting time M^W has non-negative value.
- ${\epsilon_i}^P$, ϵ^W, ϵ^C : the estimated energy consumption in the mobile device for task t_i, remote execution waiting and data communication, respectively.
- m, c, cl: the number of tasks to be executed in mobile device, the public cloud and cloudlet, respectively.

Monetary Cost Model The monetary cost represents the amount of money to run the mobile application on the proposed hybrid-MCC environment. This includes two parts: total remote task processing cost C^P in cloudlet and the public cloud and total data communication cost C^C.

$$C = C^P + C^C \tag{15}$$

$$C^P = \sum_{i}^{c} {C_i}^P \tag{16}$$

$${C_i}^P = {D_{t_i}}^P * p_{cost} \tag{17}$$

$$C^C = \sum_{i}^{n} (s_i * \beta_{cost}) \tag{18}$$

4.4 Problem Formulation

The main objective is to find the best mobile application offloading plan in which the total energy consumption on the mobile device and the total monetary cost are reduced in respect to individual task deadline and available mobile battery energy. The offloading plan represents tuple for each task t_i and the selected computation environment among local execution on the mobile device P_m, the cloudlet or the public cloud P_{cloud}. Precisely, the optimization problem is formulated as a monetary cost (C) multiplied by mobile energy consumption (E) due to the assumption of equal contribution on the optimization and the difference on both measurement units:

$$min(C * E) \tag{19}$$

Subject to:

$$D_{t_i} < \partial_i \ , \forall t_i \in A$$
$$E < e$$

5 Proposed Data-aware Offloading Technique

Mobile application offloading aims to augment mobile device capabilities by migrating computation to more powerful resources. In this work, we propose a data-aware offloading technique to study the contribution of data size for mobile application offloading decision in hybrid-MCC environment. To accomplish this objective, we adopted Particle swarm optimization (PSO) [9] as evolutionary search optimization technique to find the best offloading plan based on optimization objective in (Eq. 19). This section discuses the proposed offloading technique in which an optimized tasks allocation and scheduling plan is generated.

PSO is an evolutionary computational technique that optimizes a problem by iteratively trying to improve a candidature solution with respect to quality cost measurement. It simulates the behaviour of movement organisms in a bird flock or fish schools. In our approach, a particle represents randomized application execution plan on available resources. Fig. 2 provides an example of a particle position. There are ten tasks to schedule. In such case, a particle is ten-dimensional and its position has ten coordinates. The coordinate index (coordinate 1 through coordinate 10) maps into tasks (t_1 through t_{10}). The value of each coordinate is a real number in range [0..3). Coordinate value in range

Particle's Position

Crd.1	Crd.2	Crd.3	Crd.4	Crd.5	Crd.6	Crd.7	Crd.8	Crd.9	Crd.10
0.38	1.28	1.86	2.90	2.73	2.63	0.91	2.59	0.47	1.78

Task to Resource Mapping

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
M	CL	CL	C	C	C	M	C	M	CL

Fig. 2. Example of the encoding of a particle's position.

[0..1] correspondent task is allocated to mobile device and Coordinate value in range (1..2] correspondent task is allocated to cloudlet while Coordinate value in ranges (2..3) correspondent task is allocated to the public cloud.

To reflect the objective of scheduling tasks on the defined computation environment, the fitness function is used to determine the goodness of a particle position by estimating the optimization value for a given solution according to the total monetary cost C (Eq. 15) and the total energy E (Eq. 9) consumed by the mobile device. The fitness function accepts a PSO particle in which the position represents an application tasks scheduling solution. Energy estimation and cost calculation are based on task execution time D (Eq. 5) including pro-

Algorithm 1 Task scheduling for data-aware offloading

Inputs : set of application tasks T, computation resources R
Output : application tasks schedule S
Update resources' metadata
Setting up PSO Environment P
Initialize PSO Particles $P[NumP]$ $NumP$: number of particles
$P.gbest$ = inf Initialize global best $gbest$
for $i = 1$ to $NumP - 1$ do **do**
  Randomize $P[i].POS$
  $FitCost = PSOFitFun(P[i].POS, T)$
  UpdateBestPos $(P, P[i], FitCost)$
end for
Run PSO Iterations
for $i = 1$ to $NumL - 1$ do **do**
  for $j = 1$ to $NumP - 1$ do **do**
    Calculate $P[j].VELOCITY$
    Update $P[j].POS$
    $FitCost = PSOFitFun(P[j].POS, T)$
    UpdateBestPos $(P, P[j], FitCost)$
  end for
end for
$MinCost = PSOFitFun(P.gbest.POS, T)$
$S = (P.gbest.POS, MinCost)$
return S

cessing, communication and waiting time. The data communication time D^C for task t_i depends on task data location. D^C is only considered if data storage and computation environment are in different locations. The fitness function considers the impact of D^C time on mobile device energy and monetary cost. Algorithm 1 shows the main steps of finding the optimal offloading scheduling plan for data-aware mobile applications.

The task processing time D^P is calculated based on task t_i computation requirement and task data size. The impact of data size is measured by experimenting task processing with different data sizes. The processing time D^P is used to estimate task processing energy for local task execution, and task processing monetary cost C^P for remote execution. However, the system assumes extra energy consumption when the mobile device is in idle state, this occurs when total local execution is less than the maximum remote execution in cloudlet or public cloud.

6 Performance Evaluation

Performance evaluation for the proposed data-aware offloading technique has been conducted using two sets of experiments. The first set is a real experiment, to validate the model by comparing the result of the proposed execution model and offloading technique against real execution. For this experiment, we have implemented a real MCC environment as illustrated in Fig. 1. In the second experiment set, we used synthetic application data to evaluate the proposed offloading technique.

6.1 Experimental Setup

The configuration of the hybrid-MCC resources includes a mobile devices, cloudlets and public cloud which are listed in Table 2. We considered a single machine for task processing in the public cloud and cloudlet. For mobile network bandwidth, we conducted the experiment with three communication networks (3G, 4G, and Wi-Fi). Table 3 presents details about bandwidth values for used networks. We have recorded and used minimum and maximum bandwidth in real application execution for each network type. Recorded values are used to build a uniform distribution model for the experimental work. We initialized the application tasks structure with 30 tasks. Each task has the following properties:

Table 2. Computation Resources Configuration

Resource Type	No. Cores	Memory (GB)
EC2 Linux t2.2xlarge Intel Xeon	8	32
Cloudlet Intel Xeon	4	8
LG Nexus 5 Qualcomm	2	2

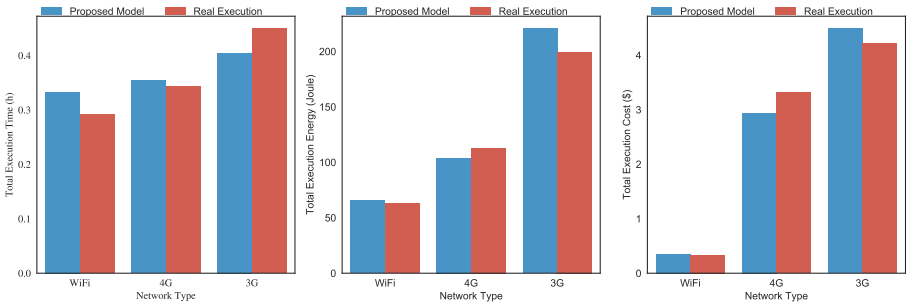
Table 3. Communication Networks bandwidth

Network Type	**Min. Bandwidth (MB/s)**	**Max. Bandwidth (MB/s)**
3G	1.5	2.6
4G	3.9	9.5
Wi-Fi	10.8	20.5
Network Latency	**Min. Latency (s)**	**Max. Latency (s)**
Latency	0.6	11.5

- Computation requirement (task workload) D^P: we used the workload model proposed by Anglano and Canonico [3]. The model used to determine the task deadline with 1000 s granularity. Task deadline values have been uniformly distributed with 205 s and 590 s for minimum and maximum time, respectively.
- The task data file locations (L) are distributed randomly between the public cloud server, the mobile device, and other cloud storage (i.e. AWS S3).
- Task data size (s) model: we assume three different scenarios for the application data model to create each task, namely, small (20-200 MB), medium (200-500 MB) and large (500-20000 MB).
- Task data sensitivity factor (w): the factor has the value between [0..1] which measures the task execution response to the change of data size.

6.2 Model Validation

In this section, we provide the validation for the proposed execution model using a real-time execution scenario. We run an application with 30-task of small data model using both proposed model and real implementation for three different communication networks.



(a) Execution Time (b) Mobile Energy (c) Monetary Cost

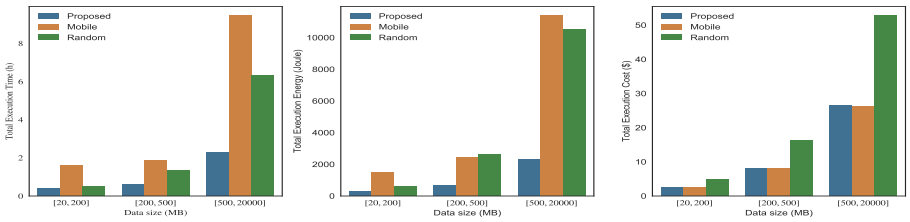
Fig. 3. Application execution model validation using real experiments for three different communication networks

Fig. 3 shows the comparison between the proposed model and real execution results in three different communication networks and three performance metrics including execution time, mobile energy and monetary cost. The result shows some discrepancies between the proposed model and real execution scenario because of the difference in available bandwidth and network latency in real execution scenario and the proposed model. We observed the major error in 3G network. As you can see in Table 3, the bandwidth for 3G network is between 1.5 and 2.6 but in real execution, the bandwidth is unstable because of bandwidth fluctuation. The average errors for execution model are 8% for execution time, 11% for energy consumption and 15% for the monetary cost. Based on the observations, the proposed execution model is accurate enough to use for offloading in the hybrid-MCC system.

6.3 System Evaluation and Experimental Results

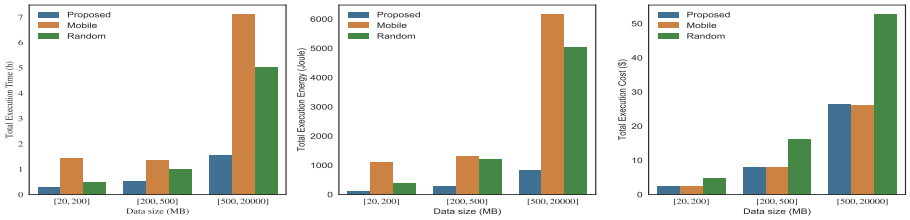
After validation of the execution model, we used the similar setup to evaluate the proposed offloading technique under different working conditions. We compared the proposed technique with two other techniques including mobile only and random offloading. The application tasks were demonstrated based on the aforementioned data size scenarios. In addition, the performance measurements that we used are execution time, mobile energy and monitory cost. Figures 4, 5 and 6 show the results of running the mobile application on three techniques for three different communication networks.

Fig. 4 shows experimental results when using the 3G network. Fig. 4 (a) illustrates that the proposed technique has significant execution time reduction in comparison to the other two techniques for the three data size models. In comparison to the random technique, execution time reduction increased from 25% in small data size to 56% in medium data size and 63% in large data size. For the same experiment, the result of the proposed technique compared to the mobile technique reduced the execution time by average 73% for three data size models. Moreover, Fig. 4 (b) shows high mobile energy consumption saving with increasing of the data size. For example, the proposed technique can save mobile energy consumption around 57% in small data size, 74% in medium data size and 78% in large data size in compared with the random technique.



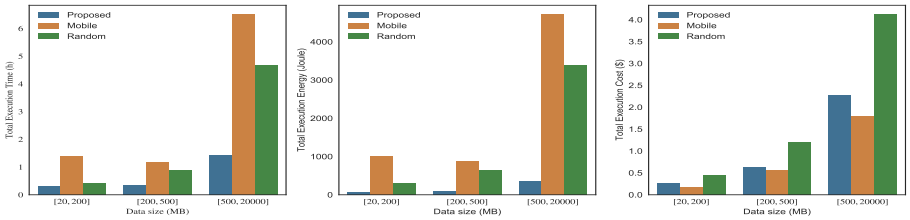
(a) Execution Time (b) Mobile Energy (c) Monetary Cost

Fig. 4. System Performance Measurements with 3G Network for different data sizes



(a) Execution Time (b) Mobile Energy (c) Monetary Cost

Fig. 5. System Performance Measurements with 4G Network for different data sizes



(a) Execution Time (b) Mobile Energy (c) Monetary Cost

Fig. 6. System Performance Measurements with Wi-Fi Network for different data sizes

For the same experiment, the result of the proposed technique compared to the mobile technique can save the mobile energy consumption by average 78% for the three data size models. Moreover, Fig. 4 (c) highlights the ability of the proposed technique to save the monetary cost around 2-3% compared to the mobile technique and cost saving of 50% compared to the random technique. This reveals the ability of the proposed technique in controlling the monetary cost as execution in the mobile environment has only communication cost while other two techniques have communication cost and cloud resources cost. In general, while the data size increasing the importance of the proposed technique to handle the data-intensive application is increasing.

Similar results can be observed in case of 4G network as shown in Fig. 5. Fig. 5 (b) result shows energy consumption saving of the proposed technique compared with the mobile technique with 89% for large data size compared to 80% using 3G network due to higher energy consumption of 4G network. Also Fig. 6 (b) shows average 90% and 84% in saving mobile energy consumption of the proposed technique compared with the mobile technique and the random technique, respectively for three data size model using Wi-Fi network. This again confirms the significance of the proposed model to save energy while using different communication networks.

Based on the results, the proposed technique improves execution time for data-intensive application for large data size sets compared to the random technique by average of 51% and saving mobile energy consumption by average of 77% while the average monetary cost saving improvement is 48%. Moreover,

the proposed technique improves execution time for data-intensive application for large data size sets compared to the mobile technique by average of 78% and saving mobile energy consumption by average of 87% while the average of monetary cost is only 11% higher due to using cloud resources and mobile communication.

In summary, the proposed technique shows high capability in handling data-intensive applications in compared with the mobile and the random techniques through reducing mobile energy consumption and monetary cost. It can be stated that for data-intensive applications, it is critical to use the proposed technique since the execution time and mobile energy consumption will be very high, but for small data size applications we could possibly use the mobile technique or the random technique.

7 Conclusions and Future Work

We proposed a QoS-aware resource allocation model to schedule data-intensive mobile applications on hybrid-MCC environment. The model generates an application execution plan with consideration to different aspects according to application complexity, input data size, available network bandwidth, and available mobile device energy. Results indicated sufficient behaviour particularly with execution time and energy. As a future work, we will study the integration of edge computing as improvement to the hybrid-MCC environment to handle data-intensive MCC application requirements. Moreover, we are planning to study task queuing behaviour on remote servers. Furthermore, we are planning to build a prediction framework for generalizing offloading decision according to most significant parameters.

References

1. Abolfazli, S., Sanaei, Z., Gani, A., Xia, F., Yang, L.T.: Rich mobile applications: genesis, taxonomy, and open issues. Journal of Network and Computer Applications **40**, 345–362 (2014)
2. Ahnn, J.H.J.: Data-Intensive Mobile Cloud Computing. Ph.D. thesis, UCLA (2015)
3. Anglano, C., Canonico, M.: Scheduling algorithms for multiple bag-of-task applications on desktop grids: A knowledge-free approach. In: Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on. pp. 1–8. IEEE (2008)
4. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Communications of the ACM **53**(4), 50–58 (2010)
5. Bangui, H., Rakrak, S., Raghay, S.: External sources for mobile computing: The state-of-the-art, challenges, and future research. In: Cloud Technologies and Applications (CloudTech), 2015 International Conference on. pp. 1–8. IEEE (2015)
6. Chun, B.G., Ihm, S., Maniatis, P., Naik, M., Patti, A.: Clonecloud: elastic execution between mobile device and cloud. In: Proceedings of the sixth conference on Computer systems. pp. 301–314. ACM (2011)

7. Cisco, C.V.N.I.: Global mobile data traffic forecast update, 2013–2018. white paper (2014)
8. Cuervo, E., Balasubramanian, A., Cho, D.k., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.: Maui: making smartphones last longer with code offload. In: Proceedings of the 8th international conference on Mobile systems, applications, and services. pp. 49–62. ACM (2010)
9. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on. pp. 39–43. IEEE (1995)
10. Giurgiu, I., Riva, O., Juric, D., Krivulev, I., Alonso, G.: Calling the cloud: enabling mobile phones as interfaces to cloud applications. In: Proceedings of the 10th ACM/IFIP/USENIX International Conference on Middleware. p. 5. Springer-Verlag New York, Inc. (2009)
11. Kemp, R., Palmer, N., Kielmann, T., Bal, H.: Cuckoo: a computation offloading framework for smartphones. In: International Conference on Mobile Computing, Applications, and Services. pp. 59–79. Springer (2010)
12. Kosta, S., Aucinas, A., Hui, P., Mortier, R., Zhang, X.: Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: Infocom, 2012 Proceedings IEEE. pp. 945–953. IEEE (2012)
13. Lin, T.Y., Lin, T.A., Hsu, C.H., King, C.T.: Context-aware decision engine for mobile cloud offloading. In: Wireless Communications and Networking Conference Workshops (WCNCW), 2013 IEEE. pp. 111–116. IEEE (2013)
14. Little, J.D.: A proof for the queuing formula: L= λ w. Operations research **9**(3), 383–387 (1961)
15. March, V., Gu, Y., Leonardi, E., Goh, G., Kirchberg, M., Lee, B.S.: μcloud: towards a new paradigm of rich mobile applications. Procedia Computer Science **5**, 618–624 (2011)
16. Nan, X., He, Y., Guan, L.: Optimal resource allocation for multimedia cloud based on queuing model. In: Multimedia signal processing (MMSP), 2011 IEEE 13th international workshop on. pp. 1–6. IEEE (2011)
17. Rahimi, M.R., Venkatasubramanian, N., Mehrotra, S., Vasilakos, A.V.: Mapcloud: mobile applications on an elastic and scalable 2-tier cloud architecture. In: Proceedings of the 2012 IEEE/ACM fifth international conference on utility and cloud computing. pp. 83–90. IEEE Computer Society (2012)
18. Sanaei, Z., Abolfazli, S., Gani, A., Shiraz, M.: Sami: Service-based arbitrated multi-tier infrastructure for mobile cloud computing. In: Communications in China Workshops (ICCC), 2012 1st IEEE International Conference on. pp. 14–19. IEEE (2012)
19. Satyanarayanan, M., Bahl, V., Caceres, R., Davies, N.: The case for vm-based cloudlets in mobile computing. IEEE pervasive Computing (2009)
20. Wang, Y., Chen, R., Wang, D.C.: A survey of mobile cloud computing applications: perspectives and challenges. Wireless Personal Communications **80**(4), 1607–1623 (2015)
21. Zhou, B., Dastjerdi, A.V., Calheiros, R., Srirama, S., Buyya, R.: mcloud: A context-aware offloading framework for heterogeneous mobile cloud. IEEE Transactions on Services Computing (2015)
22. Zhou, B., Dastjerdi, A.V., Calheiros, R.N., Srirama, S.N., Buyya, R.: A context sensitive offloading scheme for mobile cloud computing service. In: Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on. pp. 869–876. IEEE (2015)