Dynamic Partitioning and Task Scheduling for Complex Workflow Healthcare Application in Mobile Edge Cloud Architecture

Abdullah Raza Lakhan
Southeast University (China)

Dynamic Partitioning and Task Scheduling for Complex Workflow Healthcare Application in Mobile Edge Cloud Architecture

Abdullah
School of Computer and Engineering
Southeast University Nanjing China
Email: abdullah@seu.edu.cn

Dr. Prof. Li Xiaoping
School of Computer and Engineering
Southeast University Nanjing China
Email: xpli@seu.edu.cn

***Abstract* — In this paper, we are investigating the energy consumption of mobile devices and cloud resources (i.e., virtual machines) while viewing the application partitioning and task scheduling problem in the offloading system. However, offloading system enables mobile device to separate the application tasks into local execution parts and cloud server execution parts. To minimize the energy consumption of mobile devices along with cloud resources in the considered problem we have nominated a new dynamic application partitioning task scheduling (DAPTS) algorithm. Since, DAPTS aims is to efficiently dynamically partitioning the application into tasks and schedule them on the mobile device and cloud resources that one may minimize energy consumption of both mobile devices as well as cloud resources simultaneously. Experimental results show that propose DAPTS outperforms as compared to the baseline approaches.**

***Keywords*—MCA, Offloading System, DAPTS, Workflow Application, Healthcare Application.**

I. INTRODUCTION

Limited constrained mobile device cannot be performed complex workflow applications such as 3D medical-care applications due to limited processing speed, battery capacity, bandwidth utilization and storage space. However, mobile cloud computing architecture (MCA) enables mobile device to execute compute intensive tasks of an application on the cloud resources [1]. The offloading system is a technique in the MCA to which application tasks are partitioned into mobile device local execution and remaining compute intensive tasks offloaded to the cloud computing as shown in Figure 1, in order to reduce mobile device. The main costs for the offloading is the computation cost (i.e., local execution, and remote cloud execution cost), extra communication incurred by communication cost. [2]. However, present offloading studies do not reflect on with reference to the energy consumption used by cloud resources. Since, existing offloading scheme just focuses on avoiding use local CPU resources in order to save spent on processing on local devices. They paid limit attention, energy consumption due to the cloud resources while performing the offloading system. Nevertheless, the current offloading system is required adaptive environment due to user mobility, since previous proposed static offloading scheme could not stabled for application execution [3].

In this paper, we are examining the dynamic application partitioning and task scheduling in heterogeneous environment for offloading system. We have concentrated on a healthcare application in considered a problem. Whereas, a healthcare application is confined by a deadline. A healthcare application has dependent tasks and naturally can be modeled as workflow application. Our aim is to minimize total power consumption for offloading system. Whereas, total power consumption is the combination of computing (i.e., mobile CPU energy consumption, cloud resources, energy consumption) and communication energy consumption while offloading process is running along.

Despite, many efforts have been made for the current offloading system [4], [5], [6], [7] and [8], still many challenges need to be addressed in current offloading system such as:

- **Offloading allocation decision:** Before and after application partitioning and task scheduling is not trivial in heterogeneous environments (i.e., mobile devices, wireless network bandwidth, cloud resources). Existing, static scheme could not provide stable online system. A real time and the online dynamic algorithm would handle dynamic and task scheduling in heterogeneous environments.
- **Dynamic Network Connection:** Since network connection has great influence on offloading system performance [9]. It is not possible that network connection always remains stable. The network bandwidths and data sizes vary in wireless Since the network and device condition is only measured at run time, the partitioning algorithm should support adaptive changes in network and device.

With the best knowledge, dynamic optimal application partitioning and workflow task scheduling together has not been studied yet. To cope up with above challenges we have following contributions in this paper:

- In this paper, to cope with the above challenges we have proposed DAPTS algorithm framework. In which we partition the application either, which task offload or not based on mobile device current status, available network bandwidth and cloud resource available speed in order to minimize the mobile power consumption. It is adaptive, once all above parameter changes we can repartitioned the application according to new parameters. Our proposed task scheduler, schedule with all offloaded tasks on cloud resources in such way, that minimize the power consumption of all resources while performing tasks.

The rest of the paper is organized as follows. Part II tells about related work, part III, and V describes problem description and problem formulation. Proposed Algorithm is defined in section 5. Performance evaluation and conclusion are defined in part VI, and VII, respectively.

978-1-5386-8339-2/18/$31.00 ©2018 IEEE

Table I: Mathematical Notations

Notation	Description
V	Set of all tasks v
$\mathcal{V}_M$	Set of virtual machines $\mathcal{V}$
$\mathcal{V}_j$	j^{th} virtual machine in edge server
v_i	Workflow application task
W_i	Weight of the each task
D_N	Deadline of the application
ζ_j	Speed of j^th virtual machine
T_i^e	Processing time of v_i on cloud
x_{ij}	Assignment of task v_i on virtual machine j
y_{im}	Assignment of task v_i on mobile m m
B_i	Begin time of the task v_i
F_i	Finish time of the task v_i
G	Application DAG form
D_G	Application deadline
A	Most Tightly Connected Vertices
a	Arbitrary of vertex G

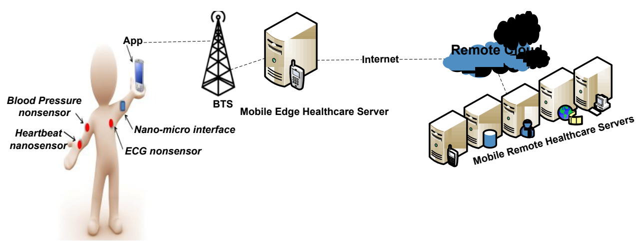

Figure 1: Mobile Cloud Architecture

II. Related Work

MCA can offer rich computing resources, mobile users can utilize these resources to alleviate the sum of computation on mobile devices and save their CPU power consumption. Consequently, MCA can save mobile energy via computation offloading. However, existing offloading schemes do not consider the power consumption of cloud resources when mobile tasks offloaded to the server for execution. A full offloading scheme is proposed in [5], which aims is to offload full application to the cloud for computation. But they didn't pay attention on communication cost which is more important for offloading system. MAUI [6], proposed partial offloading for mobile cloud application, the objective is to minimize only mobile CPU power consumption, but they did not consider cloud resource energy consumption. Cuckoo, JADE [7], [10], [8], they proposed their strategies for offloading system to minimize either mobile energy consumption or mobile device power consumption, else cloud power consumption has been ignored in current offloading system studies.

III. Problem Description

Our objective is to minimize the energy consumption of mobile device and cloud resources while performing dynamic application partitioning and task scheduling problem in heterogeneous environments. We formulate the problem with different energy consumption models such as mobile energy efficient and cloud energy efficient for mobile healthcare application. We can see energy efficient task scheduler and mobile-cloud energy models in the next sections.

A. *Energy Efficient Task Scheduler*

Our proposed task scheduler algorithm formulates subsequent suppositions:

- Each task v_i can be processed on the mobile device and cloud server.
- The network bandwidth between cloud computing resources and mobile devices is fixed and stable.
- The computing power requisite by a task v_i has a linear association with the processing time.

B. *Mobile Energy Consumption Model*

Energy consumption of mobile devices relies upon computation and communication loads [11]. To determine the energy consumption of each task v_i, we presume the task computation needs I instruction for execution. The task requires to deal with data W_i to be executed. We employ B to set for the current wireless network, the task v_i transmit $\frac{W_i}{B}$ and receive the workload. We classify the power consumption of each task v_i as follows: A workflow application task v_i has I to be executed. A task v_i has workload W_i need to be executed either on local device or cloud sever. The mobile device consumes P_c (watt per instruction) for task computation and P^{tr} (watt per second) for transmitting and receiving data [12]. A decision variable $y_{im} \in \{0,1\}$ is utilized, y_{im}=1 only if the task v_i is assigned to the mobile device m or not. The total energy consumption of mobile device for all tasks classify as follows:

$$E_{loc} = \sum_{i}^{V} P_c \times I \times y_{im} + \frac{W_i}{B}. \quad (1)$$

C. *Cloud Energy Consumption Model*

In MCA, cloud servers holds different virtual machine type, that all virtual machine instances are heterogeneous, every virtual machine (VM) has dissimilar computation speed which are illustrated as $\zeta_j = (j = 1, ..., M)$. A set of virtual machine instance can be shown by VK=$\{vk_1, ..., vk_n\}$, in which K_i^{vk} is the virtual machine assignment for task v_i. Each workflow application task has workload W_i=$\{i = 1, ..., N\}$ with deadline D_G. To minimize the power consumption of the submitted workflow tasks, we assign each application task to the lower speed VM while meeting the deadline D_G, because the lower speed VMs always leads to lower power consumption. Since a task v_i is only can be performed by one VM j, a decision variable $x_{ij} \in \{0,1\}$ is utilized, x_{ij}=1 only if the task v_i is assigned to the VM V_j. The task v_i has two execution cost (i.e., local and cloud), on cloud execution time is determined by the speed ζ_j and power consumption P_j e.g., T_i^e=$\sum_{j=1}^{\mathcal{V}} x_{ij} \times \frac{W_i}{\zeta_j} \times P_j$. The total energy consumption of all tasks on the cloud computing as follows:

$$E_{cloud} \sum_{i=1}^{V} \sum_{j=1}^{M} x_{ij} \times P_j \times T_i^e. \quad (2)$$

IV. System Model Problem Formulation

In MCA a workflow, mobile health-care application is modeled as consumption weighted Directed Acyclic Graph (DAG) i.e., G (V, E). Whereas, each task v_i is represented by a node $v_i \in V$. An edge $e(v_i, v_j) \in E$ represents communication between v_i to v_j. A task v_i could be started until associated all predecessors completed. v_1 and v_n are two dummy tasks (i.e., entry task and exist task). A task could be started until associated predecessors complete. In simple words v_j cannot be started in anticipation of v_i get the job done and $i < j$. The Mathematical Notations are marked for offloading system in Table I. The considered problem is mathematically modeled as bellow:

$$\min_{Z} = E_{loc} + E_{cloud}, \tag{3}$$

$$T_{j,0} = 0, \tag{4}$$

$$T_{j,k} = T_{j,k} - 1 + \sum_{k=1}^{V} x_{j,k} T_i^e, \tag{5}$$

$$T_i^e = \sum_{j=1}^{M} \times \frac{W_i}{\zeta_j}, \tag{6}$$

$$F_i = \sum_{j=1}^{M} T_{j,k} x_{i,j} + P_c \times I y_{im}, \tag{7}$$

$$\sum_{i=1}^{N} x_{i,j} = 1, \sum_{j=1}^{M} x_{i,j} = 1, \tag{8}$$

$$F_i \leq D_G, x_{i,j}\{0, 1\}. \tag{9}$$

The equation (3) tells about objective function. Initialization of virtual machine is zero shown in equation (4). The finish time of virtual machine $T_{j,k}$ of the virtual machine $\mathcal{V}_j$ while executing the task v_k is determined by the finish time of previous task $v_k - 1$ and the execution time $\sum_{i=1}^{V} x_{k,j} T_k^e$ of the current task v_k and the $t_{j,k}$ is formulated in the Equation (5) and T_i^e is defined in equation (6). Equation (7) shows the total finish time of all tasks on cloud virtual machines and mobile device. Assignment of each task on either on each virtual machine or mobile device is shown in equation (8). The last equation (9) shows that all tasks must be completed before application given deadline.

V. Proposed Algorithm Application Partitioning

Proposed algorithm DAPTS has two sub problems such as dynamic application partitioning and task scheduling as shown in Algorithm 1. A dynamic application partitioning can be done based on task size, mobile device, network bandwidth and cloud resources factors. We formulated the dynamic application problem based on the min-cut algorithm [13]. Application partition into local execution and remote execution should be based on Algorithm 2. In Algorithm 2 there are three main functions such as merging, min-cut phase and min-cut function [14]. Algorithm 2, always return optimal partitioning of application in order to minimize the mobile energy consumption. In order to minimize the energy consumption of cloud server, we have proposed energy efficient task scheduling algorithm as shown in Algorithm 3. Task scheduler Algorithm 3 search all virtual machines which can be capable to execute all offloaded tasks with lower power consumption within application deadline. In algorithm 2, we perform all functions for example merging function, min-cut phase function and min-cut function in order to dynamically partition the application according to available parameters (i.e., network bandwidth, task size).

Algorithm 1: DAPTS Framework

Input : $v_i \in V$;
Output: $\min_Z$;
begin
  $Z \leftarrow 0$;
  Application Partitioning based on equation (1) (2) Workflow Task Sequence;
  foreach $v_i \in V$ **do**
    Optimal VM Searching ;
  Task Sequence Adjustment;
  return $Z*$;

Algorithm 2: Min-Cut Function

Input : (G, w);
begin
  $w_t(\min_{cut} \Leftarrow \infty$;
  for *(G'=1)* **do**
    (G', w_t)= merging(G', w_t);
    while $(v \in V) > 1$ **do**
      cut$(A - t, t)$;
      s,t=$\min_{cut}$ phase(G', w_t);
      if $w_t(c_{cut}(A - t, t)) < w_t(\min_{cut})$ **then**
        $\min_{cut} \Leftarrow c_{cut}(A - t, t)$;
    merging(G', w_t, s, t);
  return $\min_{cut}$, $\min_{cut}$ grouping list;

A. *Task Scheduling Algorithm*

In this section, we schedule all tasks on cloud virtual machines. Algorithm3 search optimal virtual machines in which all tasks could be completed within application deadline with lower power consumption.

$$T_i^{slack} = D_i - F_i \tag{10}$$

$$\overline{F}_i = \sum T_i^e \frac{\sum_{j=1}^{M} W_i}{\sum_{j=1}^{M} \zeta_i} \tag{11}$$

Tasks adjustment can be done on the following way:

- Shortest deadline First (SDF): We sort the set of workflow applications based on their deadline.
- Shortest slack time first (SSF): The application tasks are sorts according to the task slack time (TST).
- Shortest weight First (SWF): The applications are sequenced based on the weight of all tasks.

Based on equation (10) and (11), we can determine the finish time of all tasks on virtual machine after scheduling.

$$\zeta_{j^*} = \frac{W_i}{\zeta_j}. \tag{12}$$

In Algorithm 3, line 2, VMs are sorted by calculating P_j with the descending order, and put into Q_{vm} in which VMs are iteratively traversed. In line 3, initially all VMs are null. The available time $T_{j,0}$ of each VM in the Q_{vm} is initialized to 0. Line 7 to 11, if the available time of the VM $\mathcal{V}$ plus the execution time of vi is less than the deadline D_G. v_i is assigned to the VM $\mathcal{V}$, and the new available time $T_{j,i}$ of $\mathcal{V}$

Algorithm 3: Optimal VM Searching

Input: ($v_i \in V$) to Scheduling;

begin

  $Q_{vm} \leftarrow$ Sort all VMs by their speed ζ_j^* based on equation (12) in ascending order;

  $\mathcal{V} \leftarrow$Null;

  foreach $\mathcal{V}_j \in Q_{vm}$ **do**

    $T_{j,0} \leftarrow 0$;

  foreach $\mathcal{V}_j \in Q_{vm}$ **do**

    Calculate T_i^e of $\mathcal{V}_j$ based on equation (5);

    if $T_{j,i-1}$+ $T_i^e \leq D_G$ **then**

      Calculate the $T_{j,i}$ of $\mathcal{V}_j$ by equation (6);

      $\mathcal{V} \leftarrow \mathcal{V}_j$;

      Schedule all tasks $v_i \in V$ based on equation (10) and (11);

      break;

    Optimize total power consumption based on equation(3) $Z^*(5)$;

  return $Z^*, \mathcal{V}$

is dynamically updated. The VMs are sorted in the Algorithm 2, the VMs are swapped at least $M \times log(M)$ times. Besides, the traverse of the sorted VMs consumes M times, therefore, the time complexity of the Algorithm 3 is O($M \times log(M)$).

VI. Performance Evaluation

In this paper, we are doing simulation on two sub problems such application partitioning and task scheduling. For application partitioning, we are taking call graph as a input of an application (i.e., healthcare application), and via application partitioning algorithm via energy Efficient Task Scheduler we schedule all tasks on the mobile device and cloud resources in such a way that minimize the objective function.

A. *Effectiveness of Proposed Algorithm*

To determine the efficiency of the proposed DAPTS algorithm, the following fixed values must be known in advance. For example, **fixed values:** these values are closely related to power consumption P_m, P_i, and P_{tr} and to the specific mobile device. The configuration we have employed such as PDA HP IPAQ with large Intel-Scale processor by subsequent values: $P_{tr} \approx 1.3W$, $P_m \approx 0.3W$ and $P_i \approx 0.7W$. **Precise values:** these parameters are closely related data transfer size, network upload and download and current mobile and cloud speed factor F. **Fluctuation values:** these values are scrupulously to mobile device current workload status, network status and cloud status. Due to adaption and fluctuation, it is not trivial to calculate this value initially. The profiling (program (i.e., mobile workload and other status) and network (i.e., bandwidth and available 3G/4G, WIFI and etc) can be enabled to calculate these values during fluctuation and adaption. Simulation setup again divides the application workload into mobile execution and remote execution, and then schedule related tasks along their respective processor. All simulator parameters are explained in Table II. Energy efficient task scheduling is not trivial in heterogeneous environments. For task scheduling, algorithm values should calibrate the components and parameters, and workflow applications are generated randomly. Healthcare workflow applications are created with different five sizes such as $Q_w \in \{20, 40, 60, 80, 100\}$. Since, each healthcare workflow application is comprised of four unlike figures of tasks i.e., $Q_t \in \{50, 100, 200, 500\}$. We produce ten combinations of Q_w and Q_t respectively. However, each healthcare workflow application is bounded by deadline. The deadline of each workflow D_w is expressed as follows:

Table II: Simulation Parameters

Simulation Parameters	Values
Languages	JAVA
Mobile Phones	Android ARM Processor
Cloud Processors	X86 architecture
Application Fine-grained	Tasks
Offloading Type	Dynamic
Offloading Parameters	Network Bandwidth
Offloading Parameters	Mobile-Cloud Speedup factor
Simulation Time	6 hours
Experiment Repetition	14
No. of Mobile devices	100 to 1000
Mobile CPU power consumption P_c	$\approx 0.7W$
Power Consumption Transfer data P^{tr}	$\approx 0.7W$
WAN-WLAN Network Bandwidth	20 to 300 mbps
Standard task size	1500 to 2000 MI
Upload/download data size	2000/150 KB
Possibility offload to remote cloud	80%
VM processing speed cloudlet/cloud	1200/22000 MIPS
VMs speed	500-2500 MIPS
VMs RAM	2GB-4GB
CPU-Utilization cloudlet/cloud	15∼0

$$D_w = T_w^{f_i} + \gamma + T_w^{f_i}, \tag{13}$$

$T_w^{f_i}$ is the workflow of tasks with finish time, whereas f_i is calculated based on equation (15). A γ is described as a factor to control the tightness of the deadline D_w, and $\gamma \in \{0.2, 0.4, 0.6, 0.8, 1\}$. Hence, each healthcare workflow application exactly has diverse deadline i.e., $\{D_1, D_2, D_3, D_4, D_5\}$. To evaluate the performance of the proposed DAPTS algorithm next to healthcare heuristics based DEA benchmark [15] and [16] to verify the strength of the proposed algorithm. The calibration parameters of tasks are same as a healthcare workflow, the performance evaluation of the DAPTS is measured at diverse deadlines since strict to lose. The DAPTS has RPD (Relative-Percentage-Division) is utilized to compare with existing schemes such as non-offloading and described as follow:

$$RPD\% = \frac{Z - Z^*}{Z} \times 100\%, \tag{14}$$

total save energy is equal to Z our objective function and Z^* is the baseline approaches such as [4], [5], [6], [7] and [8]. $RPD\%$ shows that proposed schemes is save more energy as compared to baseline approaches.

B. *Algorithm Comparison*

Existing offloading schemes algorithms such as full offloading (FUL) [5], non-offloading (NOF) [7], [10], partial offloading (PAR) [6] is compared with proposed DAPTS based algorithm components. As we described above in DAPTS has six components likewise, Merging function, min-cut phase and min-cut, SDF, ST, and SWF respectively. We

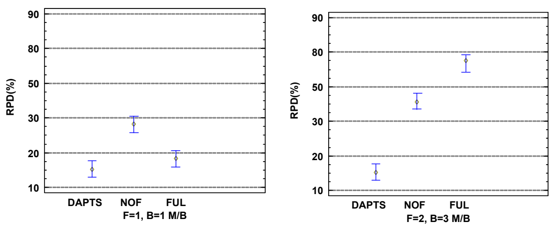

Figure 2: Application Partitioning Performance based on speed up factor F and bandwidth B

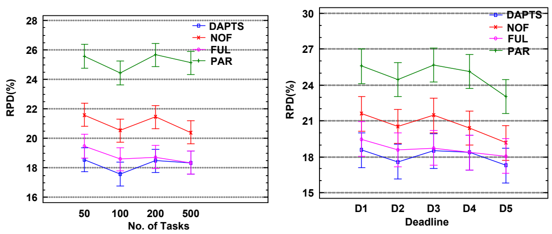

Figure 3: Workflow Application no. of tasks completed within a given deadline

evaluated the overall performance and validity based on above components, and all RPD results of proposed algorithm are better as compared to all benchmark heuristics bounded by deadline constraints [15] and [16]. According to Figure 2 proposed algorithm RPD% is better on different speedup factor F and B bandwidth than existing heuristic techniques. Figure 3 shows that healthcare workflow application tasks are completed within a given deadline after partitioning. The RPD% of proposed algorithm DAPTS is optimal in all workflow applications as shown in Figure 4. We have done experiment on different speed-up factors and bandwidths, then Figure 5 shows proposed algorithm DAPTS is optimal and consumes lower energy consumption as compared to others.

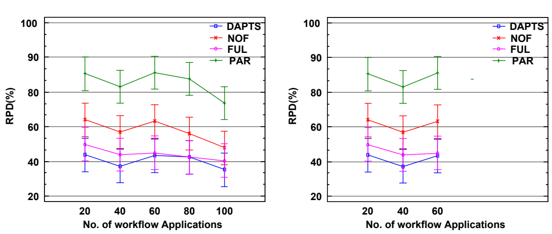

Figure 4: Multiple Workflow Applications

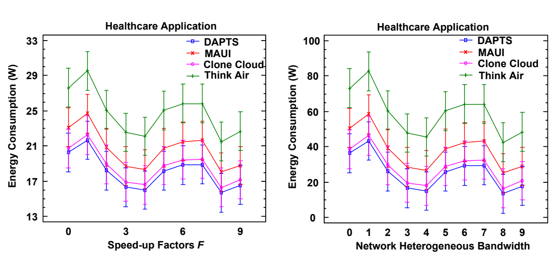

Figure 5: Energy Consumption In Heterogeneous Environments

VII. Conclusion

In this paper, we have proposed dynamic application partitioning (DAPTS) algorithm and task scheduling for real time healthcare application. We have evaluated our proposed algorithm in the healthcare benchmark workflow application and comparison with benchmark heuristics, and finish all workflow tasks within a given deadline. In this paper, our goal is to mobile power consumption as well as cloud resources while performing adaptive application partitioning and task scheduling.

References

[1] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *arXiv preprint arXiv:1712.00030*, 2017.

[2] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 175–186, 2015.

[3] H.-S. Lee and J.-W. Lee, "Task offloading in heterogeneous mobile cloud computing: Modeling, analysis, and cloudlet deployment," *IEEE Access*, 2018.

[4] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1294–1313, 2013.

[5] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *Proceedings of the sixth conference on Computer systems*. ACM, 2011, pp. 301–314.

[6] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, 2010, pp. 49–62.

[7] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," in *International Conference on Mobile Computing, Applications, and Services*. Springer, 2010, pp. 59–79.

[8] K. Akherfi, M. Gerndt, and H. Harroud, "Mobile cloud computing for computation offloading: Issues and challenges," *Applied Computing and Informatics*, 2016.

[9] M. Shiraz, A. Gani, A. Shamim, S. Khan, and R. W. Ahmad, "Energy efficient computational offloading framework for mobile cloud computing," *grid computing*, vol. 13, no. 1, pp. 1–18, 2015.

[10] H. Qian and D. Andresen, "Jade: An efficient energy-aware computation offloading system with heterogeneous network interface bonding for ad-hoc networked mobile devices," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on*. IEEE, 2014, pp. 1–8.

[11] S. Wu, C. Niu, J. Rao, H. Jin, and X. Dai, "Container-based cloud platform for mobile computation offloading," in *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*. IEEE, 2017, pp. 123–132.

[12] M. A. Hassan, K. Bhattarai, Q. Wei, and S. Chen, "Pomac: properly offloading mobile applications to clouds," in *HotCloud'14 Proceedings of the 6th USENIX conference on Hot Topics in Cloud Computing*, 2014, pp. 7–7.

[13] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A context sensitive offloading scheme for mobile cloud computing service," in *2015 IEEE 8th International Conference on Cloud Computing*, 2015, pp. 869–876.

[14] Y. Ye, H. Zhang, X. Xiong, and Y. Liu, "Dynamic min-cut clustering for energy savings in ultra-dense networks," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, 2015, pp. 1–5.

[15] A. Ettorchi-Tardy, M. Levif, and P. Michel, "Benchmarking: A method for continuous quality improvement in health," *Health Policy*, vol. 7, no. 4, 2012.

[16] C. K. Kane and D. W. Emmons, "New data on physician practice arrangements: Private practice remains strong despite shifts toward hospital employment," 2013.