Mobile Cloud-Based Big Healthcare Data Processing in Smart Cities

Md. Mofijul Islam, Md. Abdur Razzaque, *Senior Member, IEEE*, Mohammad Mehedi Hassan, *Member, IEEE,* Walaa Nagy, Biao Song, *Member, IEEE*

***Abstract*—In recent years, the Smart City concept has become popular for its promise to improve the quality of life of urban citizens. The concept involves multiple disciplines, such as Smart healthcare, Smart transportation, and Smart community. Most services in Smart Cities, especially in the Smart healthcare domain, require the real-time sharing, processing and analyzing of Big Healthcare Data for intelligent decision-making. Therefore, a strong wireless and mobile communication infrastructure is necessary to connect and access Smart healthcare services, people, and sensors seamlessly anywhere at any time. In this scenario, mobile cloud computing (MCC) can play a vital role by offloading Big Healthcare Data related tasks such as sharing, processing and analysis, from mobile applications to cloud resources, ensuring quality of service (QoS) demands of end-users. Such resource migration, which is also termed Virtual Machine (VM) migration, is very effective in the Smart healthcare scenario in Smart Cities. In this paper, we propose an Ant Colony Optimization (ACO) based joint VM migration model for a heterogeneous, MCC-based Smart Healthcare system in Smart City environment. In this model, the user's mobility and provisioned VM resources in the cloud address the VM migration problem. We also present a thorough performance evaluation to investigate the effectiveness of our proposed model compared to state-of-the-art approaches.**

***Index Terms*—Smart Healthcare, Smart City, Big Data, Quality of Service(QoS), Virtual Machine Migration, Ant Colony Optimization.**

I. Introduction

Due to recent advancements in Information and Communication Technology, the Smart City concept has become an excellent opportunity to improve the quality of everyday urban life activities [2]. By connecting Smart objects, people, and sensors various services can be provided, such as Smart healthcare, Smart transportation, and Smart community [3]. Most Smart City services, especially emerging Smart healthcare services, demand anywhere, anytime real-time computation. Critical patient monitoring, telemedicine, patient data collection, and personalized medical services [11] [35] [40] [17] are major applications in this domain. These healthcare services and applications generate copious amounts of Big Healthcare Data in real-time, thus requiring computational resources to be made available nearby [9]. However, without a strong wireless and mobile communication infrastructure, it is difficult to connect and access computational resources for processing, sharing, and analyzing of Big Healthcare Data with minimum latency [27] [43].

The emergence of mobile cloud computing (MCC) [10], [25], [34] facilitates reduction of task-execution time and real-time communication latency for such Smart Healthcare applications in the Smart City environment [12] [7] [37] [28] [1]. MCC effectively utilizes distributed cloud server resources such as CPU, memory, network, and ports to execute the mobile Smart healthcare applications. Using MCC, mobile devices (MD) can offload these applications to a resourceful cloud server for faster execution [6], [8], [13], [16]. Moreover, MCC allows real-time query processing in Smart healthcare applications that is vital for patients life. However, the long distance between cloud servers and the MDs may increase the response time for interactive applications, increasing the total execution time. To alleviate this problem, a *cloudlet* is proposed in [36], which is a resourceful local cloud that brings remote cloud resources closer to the mobile user. By offloading tasks to the nearest cloudlet, the user can decrease total task execution time.

Virtualization technology introduces a middle layer between the hardware and software layers in a cloudlet, allowing the hardware resources to be shared by means of VM. Resources (e.g., CPU, memory, network bandwidth, etc.) in a cloudlet are provisioned to these VMs. Resource provisioning in cloud computing is a well-studied area [4], [30], [39], [41]. However, the mobility in MCC introduces several challenges to maintain an acceptable Quality of Service (QoS) when provisioning cloud resources. Mobile users may move from one Access Point (AP) to another, increasing their distances between current locations and the cloudlet, where the tasks are provisioned. This increases the task-execution time. To address this issue, we propose a VM migration technique for a heterogeneous MCC system following the user's mobility pattern. That is, when a user moves from one cloudlet to another cloudlet, the resource or VM must be migrated to the cloudlet that is nearest to the user.

Consider the following scenario: a blind user is executing an application that takes an image from his surroundings. Then, the application processes the image in the cloudlet and gives a response to the user's local client. That is, the application continuously uploads some data and the cloud server processes this data to provide responses back to the user. Now, if the blind user moves away from the current cloudlet, then he

This work was supported by the Deanship of Scientific Research at King Saud University through Research Group No. RGP-281)

Md. Mofijul Islam and Md. Abdur Razzaque are with the Department of Computer Science and Engineering, University of Dhaka, Bangladesh, e-mail: akash.cse.du@gmail.com, razzaque@du.ac.bd

Mohammad Mehedi Hassan, Walaa Nagy and Biao Song are with the Information Systems Department, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia, e-mail: {mmhassan, 435204202, bsong}@ksu.edu.sa

or she will experience a delayed response from the mobile application executing in the cloudlet, degrading the overall performance of the application. To avoid this performance degradation, it is necessary for the system to adopt a VM migration method to choose a cloudlet that is currently closer to the user to which to migrate the VM.

User mobility is not the only reason forcing a VM to migrate. Migration can be initiated to minimize the over-provisioned resources and thus improve the overall system objectives. For instance, if a VM is required to be migrated from a cloudlet to any of the candidate cloudlets, the new cloudlet may not have the same type of VM. In that case, a VM with more resource than the current one must be chosen and provisioned in order to migrate the VM and thus minimize task-execution time. However, in this approach, the VM migration is provisioned more resources than the required. Therefore, this over-provisioned resources greatly decreases the system objectives, as it reduces the number of provisioned VMs in the cloudlets. Furthermore, the joint VM migration approach, where a set of VMs is remapped based on the VM task execution time and over-provisioned resources, can help to effectively increases the overall system objectives. In contrast to the joint VM migration approach, single VM migration can only improve a particular user objectives but not the system objectives.

In the literature, some researches have been conducted on VM migration in MCC [14] [31], [38], [42]. However, these methods are more suited for static task-execution. Furthermore, most state-of-the-art works have only considered a single VM migration approach. Though, this can help to improve a particular user's objectives, it can negatively impact not only the cloudlet system but also other users. Therefore, a joint VM migration model that effectively considers user mobility and the over-provisioned resources of the cloudlet has yet to be proposed in the literature.

In this work, we propose a VM migration (VMM) model based not only on user mobility but also on load of cloudlet resources. The objective is to select the optimal cloud server for a mobile VM in addition to minimizing the total number of VM migrations, reducing task-execution time. We use Ant Colony Optimization (ACO) to identify the optimal target cloudlet. The main contributions of our work are stated bellow:

- We develop an Ant Colony Optimization (ACO)-based VM migration model, in which VM are migrated to candidate cloud servers so as to maximize the total utility of the MCC system.
- Mobility-aware selection of cloudlets for VM provisioning in our proposed PRIMIO system helps significantly to reduce service provisioning time.
- We introduce a joint VM migration approach to optimize both the resource utilization and task execution time, diminishing the shortcomings of a single VM migration approach.
- The results of performance evaluation, depicted from test-bed implementation and extensive experiments, show that the proposed PRIMIO system achieves significant improvements compared to state-of-the-art works.

The remaining paper is organized as follows. In Section II, we discuss on the state-of-the-art work on VM or task migration in MCC. Next, in Section III, we present the network model and assumptions. After that, in Section IV, we present the problem formulation of our VM migration model. In section V-B, we present a meta-heuristic ACO-based VM migration solution for jointly migrating a set of VMs. Subsequently, in Section VI, we present a numerical evaluation of our proposed model to investigate it's effectiveness of our proposed model compared to the state-of-the-art. Finally, we draw conclusions and mention directions for our future works in Section VII.

II. Related Work

With the advent of the smart City, smart healthcare services are emerging to improve urban citizens' quality of life [21] [23] [22] [18] [19] [20]. However, to connect and access smart healthcare services, people, and sensors, seamlessly, anywhere and any time, MCC plays a vital role. In MCC services, users can offload (-part of) a task in cloudlet for faster execution. Existing literature discusses several methodologies to detect when a client should offload task in a cloudlet [4], [6], [8], [13], [30], [39], [41]. However, little works has been conducted in the field of VM migration in MCC systems. A traffic-aware, cross-site VM migration model is proposed in [31]. In this model, when multiple VMs require migration, an arbitrary sequence of VM migration congests the bandwidth of inter-site links, thus reducing the number of successful VM migrations. Then, the VM migration problem is formulated as a Mixed Integer Linear Programming (MILP) problem and a heuristic algorithm is used to get an approximate optimal result.

Besides this, a mobility induced service migration for MCC is proposed in [42]. In this work, a threshold-based optimal service migration policy is developed, where the VM migration problem is modeled as a Markov decision process (MDP). This proposed system considered mobile users to follow a one-dimensional asymmetric random walk mobility model. A service is migrated from one micro-cloud to another when a user is in states bounded by a set of predefined thresholds.

In addition, three lightweight task-migration models are developed in [14]:

- *Cloud-wide task migration* where the task-migration decision is made by a central cloud, which maximizes the objectives of a cloud provider.
- *Server-centric task migration* where all migration decisions are made by the server, where the task is currently executing .
- *Task-based migration* where migration is initiated by the task itself.

In this approach, the migration decision is made after each decision epoch, based on user's mobility and remaining task-execution time. This proposed method considers the *increasing data volume transfer time* during task migration from one cloud to another.

Meanwhile, in our previous work, we proposed a mobility- and load-aware Genetic Algorithm-based VM migration approach, GAVMM [24], to minimize task-execution time. However, this approach mainly tries to minimize the provisioned task execution time without considering over-provisioned resources in the cloudlets. Thus, the GAVMM fails to minimize resource over-provisioning in the cloudlets.

However, all state-of-the-art works use the single VM migration approach, where a cloudlet migrates a single VM to another cloudlet. This approach relaxes the problem formulation but fails to effectively optimize the whole-system objectives. Instead, we here use a joint VM migration, where a set of VMs jointly migrates to a set of coudlets, allowing effective optimization of resource usage and task-execution time.

In summary, most VM migration methodologies do not effectively consider user mobility alongside load condition of cloudlet servers in a heterogeneous MCC system. This increases service downtime, especially for those applications where the user frequently interacts with the provisioned cloudlet. In addition, when migrating a VM, over-provisioned resources in the target cloudlet must also be considered; otherwise, the total number of provisioned VMs in the target cloudlet will greatly be reduced. To the best of our knowledge, this work is the first to efficiently utilize ACO system to develop a VM migration approach for minimizing the task-execution time and optimizing the cloudlets resource usage. Moreover, we extend the VM migration model to jointly migrate a set of VMs to a set of cloudlets in order to minimize task-execution time and to minimize resource over-provisioning compared to single VM-based migration approaches.

III. Mobile Cloud System Architecture

We assume a three-tier Mobile Cloud Computing (MCC) environment, where a set of M access points (APs) comprise the backbone network. Tier one represents the master cloud, which consists of several public cloud providers, such as Google App Engine, and Microsoft Azure Amazon EC2. A set of high-speed interconnected cloudlets constitute the tier two or the backbone layer of the mobile cloud architecture. Smartphones, wearable devices or other mobile devices constitute the tier three or user layer. Users access the nearest cloud resources using devices from tier three. Each AP is connected to any of C cloudlets, denoted as $C = \{C_1, C_2, C_3 ....., C_L\}$. We present a sample network scenario in Fig. 1. These cloudlets are connected using the backhaul network where bandwidth between cloudlets i and j is denoted as $B_{i,j}$.

A set of cloudlets is controlled and monitored by the master cloud (MC). All cloudlets route their hypervisor information to the master clouds and they are connected to the MC with a high-speed network connection. A cloudlet server i has fixed processing power S_k^p and memory M_k. Each cloudlet provisions N number of VMs, denoted as $V = \{V_1, V_2, V_3 ....., V_N\}$. Users can offload their tasks to a dedicated VM. We further assume that each user is mobile and execute a task in different VMs over the task lifetime. We assume that there are no inter-dependencies among those VMs. In the remainder of this paper, we use task and VM interchangeably.

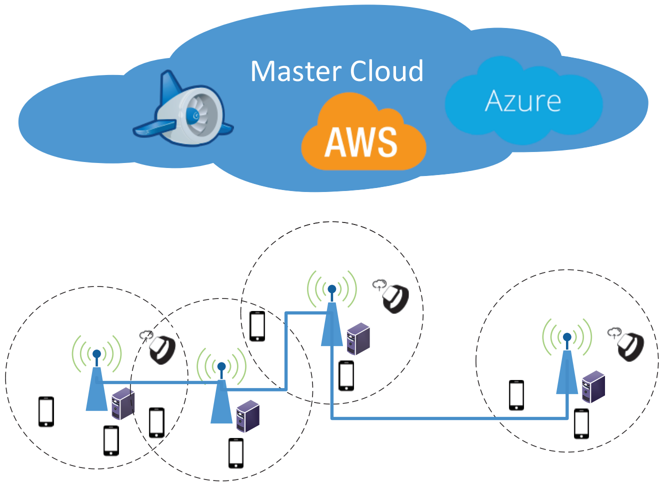


Fig. 1. Mobile Cloud Architecture

A. System Architecture

B. Virtual Machine Migration System

In Fig.(2), a computation migration system is depicted, which has two important functionality. First, the computation migration between resource-constrained mobile devices and cloudlets; Second, virtual machine migration between two cloudlets to minimize the task execution time as well as resource over-provisioning. The application offloading manager helps a mobile device to decide which part of the mobile application should be offloaded and where [6], [8], [26], [30]. The client-request handler in a cloudlet manages incoming task execution requests from mobile devices and dispatches to the virtual machine scheduler. The VM scheduler uses the information from hypervisor to initiate the VM scheduling. Moreover, after a specific time epoch, the VM migration manager interacts with the VM scheduler to select tasks and to migrate so as to reduce the task-execution time. In this work, our main concern is to develop an algorithm for the VM migration manager for selecting a set of tasks and cloudlets, remapping tasks to minimize execution time. In the VM migration manager, there are three main components- VM Selection, Joint VM Migration Engine, and VM Migration Initiator, which contribute to making optimal migration decisions. Moreover, the user mobility manager takes local mobility management information in conjunction with global mobility prediction manager to decide on a set of probable cloudlets for a user. Finally, the VM migration manager initiates migration process for a set of tasks requiring faster execution.

C. Assumptions

We assume that a cloudlet's VM scheduler determines the amount of resources to allocate to a certain VM during its creation and that it is allowed to update the VM size during scheduling intervals, if deemed necessary in support of meeting user QoS or increasing resource utilization [5] [33]. We also assume that the pre-copy live VM migration method [29] is used by the cloudlets, which allows migration of internal state, memory and application data associated with a VM from its provisioned cloudlet to a destination one.

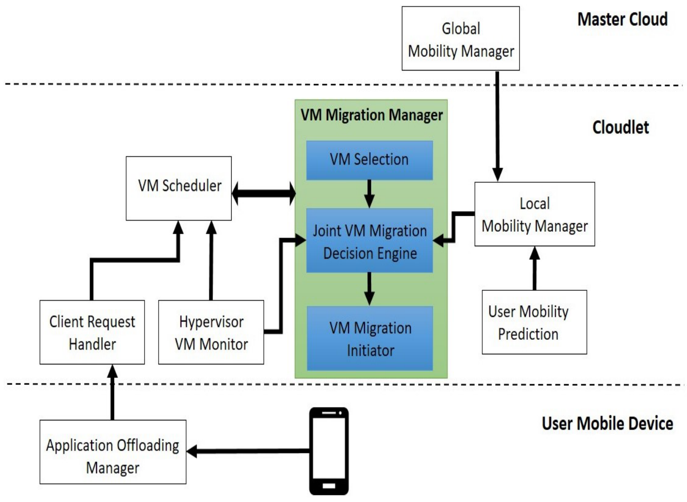


Fig. 2. Virtual machine migration system

Moreover, in this work, we assume that users will have walking mobility speeds. We employ a state-of-the-art user mobility prediction approach, ENDA [30], to determine the predicted cloudlets set for each provisioned VM. However, the problem of thrashing might appear for users with high mobility speeds (e.g., vehicular mobility), which we do not consider in this work.

TABLE I
NOTATIONS

Notations	Description
V^s	Candidate selected VM set for migration.
$V^c_{j,u}$	Candidate VM set for VM c in cloudlet j.
$T_{u,v}$	Task completion time including the task transfer time if migrate from VM u to v.
$T^e_{u,v}$	Task execution time if migrate from VM u to v.
$T^e_{u,v}$	Task transfer time between VM u and v.
T^d_u	Task deadline which is executing in VM u.
α	Task type weight parameter in a system.
$R^0_{u,v}$	Resource over-provisioned when VM u migrate to v.
$T^{p,r}_u$	Required task execution time penalty for task, which is executing in VM u.
$T^{p,d}_u$	Task execution time deadline penalty for task, which is executing in VM u.
T^u_r	Required task execution time for a task, which is executing in VM u.
$\beta^u_{d,r}$	User application preference on attribute 'd' and 'r'.
τ_0	Initial pheromone value.
$p^z_{u,v}$	VM u choosing probability over VM v in ant z.
γ_l	Local pheromone update weighted parameter.
γ_g	Global pheromone update weighted parameter.
$\Delta\tau_{u,v}$	Global pheromone value for the all VM u, v pair in global solution.
$\eta_{u,v}$	Local heuristic value when migrate VM u to v.
$\eta^R_{u,v}$	Local heuristic value of resource over-provisioned when migrate VM u to v.
$\eta^E_{u,v}$	Local heuristic value of task completion time when migrate VM u to v.

IV. Multi-objective Problem formulation

Mobile Cloud Computing (MCC), mobile devices offload the most resource intensive applications such as image processing applications, augmented reality applications etc., to the nearest cloudlet to minimize total task-execution time. However, when a mobile user moves from one cloudlet to another, the VM must be migrated and provisioned into another cloudlet to minimize the interaction time between the mobile user and the provisioned VM. The resulting total execution time helps to meet the task deadline. When a VM is selected for migration, a cloudlet must choose that VM in such a way to also minimize the resource over-provisioning. Therefore, a VM must be mapped to another cloudlet VM. There are several approaches to map and provisioned a VM to that of new cloudlet. Migrating a single VM based on the preferences of a particular mobile user can help maximize the objectives of just one user, but it fails to improve the overall system objective and also can have negative impact on other user's objectives. Therefore, in our proposed approach, we jointly consider all VMs in a cloudlet remapping those to a set of cloudlets, if necessary.

In the proposed VM-migration problem, both the total task execution time and amount of over-provisioned resources in the cloudlets must be minimized. Thus, this is a multi-objectives problem, where a set of VMs V are remapped to a set of cloudlets C. The proposed VM-migration problem has two objectives: (1) to minimize total task-execution time and (2) to minimize the resources wasted by the cloudlets. We can formulate our multi-objective VM migration problem as follows,

Minimize:

$$Z = \sum_{u \in V^s_i} \sum_{v \in V^c_{j,u}} \left\{ \alpha \times T_{u,v} + (1-\alpha) \times R^{o,i}_{u,v} \right\} \times x_{u,v} \tag{1}$$

s.t.

$$\sum_{v \in V^c_{j,u}} x_{u,v} \le 1 \qquad \forall u \in V^s_i \tag{2}$$

$$\sum_{u \in V^s_i} x_{u,v} \ge 0 \qquad \forall v \in V^c_{j,u} \tag{3}$$

$$(u^r \le v^r) \qquad \forall r \in R^u_i, u \in V^s_i, v \in V^c_{j,u} \tag{4}$$

$$T_{u,v} \times T^{max}_{u,v} \times x_{u,v} \le T^d_u \qquad \forall u \in V^s_i, v \in V^c_{j,u} \tag{5}$$

where, the objectives in Eq.(1) represent the minimization of the total task-execution time of provisioned VMs and the minimization of resource over-provisioning in the cloudlets. $x_{u,v}$ is a binary variable which will be 1 when the VM u in cloudlet i is provisioned to VM v in cloudlet j; α is the weight parameter of application-types, defining how much weight each application has in reducing task-completion time or in optimizing system resources. $T_{u,v}$ is the normalized total task-execution time when VM u in cloudlet i was provisioned to VM v in cloudlet j, which can be defined as follows,

$$T_{u,v} = \frac{(T^e_{u,v} + T^t_{u,v})}{T^{max}_{u,v}} \tag{6}$$

$$T^{max}_{u,v} = \max_{b \in V^c_{j,u}} \left(T^e_{u,b} + T^t_{u,b} \right) \tag{7}$$

where $T^{e}_{u,v}$ is the task execution time if the VM u is migrated and provisioned to VM v and $T^{t}_{i,j}$ denotes the VM transfer time from cloudlet i to cloudlet j. $T^{t}_{i,j}$ can be defined as follows,

$$T^{t}_{u,v} = \frac{D_u}{B_{u,v}} + T^{q,d}_{u,v} \tag{8}$$

here, D_u is the data associated with the VM u, which includes both VM state and application data; $B_{u,v}$ denotes the bandwidth of the link between the cloudlets running VMs u and v. In addition, $T^{q,d}_{u,v}$ denotes the VM execution queuing time for the VM u in VM v. In the proposed VM-migration approach, the full VM image is not migrated, but rather only the data which are generated by the provisioned VM. Because, in our VM-migration approach, the application in the migrated VM is stopped and later resumed on the migrated cloudlet upon completion of the migration operation.

In Eq.(1), $R^{o,i}_{u,v}$ denotes the normalized over-provisioned resource, when a VM u of cloudlet i is migrated to a VM v. It can be defined as follows,

$$R^{o,i}_{u,v} = \frac{1}{|R^{u}_{i}|} \sum_{r \in R_u} \frac{v^r - u^r}{u^r} \tag{9}$$

where, R^{u}_{i} are the resources required by the VM u in the cloudlet i. In addition u^r and v^r are the resourcea of type r, provided by the VMs u and u, respectively.

Constraint (2) ensures that one VM is provisioned or migrated to one and only one VM in a particular cloudlet. In addition, constraint (4) ensures that when the system migrates a VM u to a new VM v in another cloudlet, the new VM v has at least the same amount of resources in currently provisioned VM in the cloudlet i. Furthermore, constraint (5) ensures that a VM can be migrated to a cloudlet where the task execution deadline will be met.

V. Meta-Heuristic Virtual Machine Migration

Not all VMs in a cloudlet provisioned mobile-device applications require migration to meet the user task-execution deadlines or to increase the system objectives. Rather, migration of a subset of VMs could improve user experiences. Therefore, as a first step, the proposed PRIMIO, **PR**ioritized meta-heur**I**stic virtual **M**achine migrat**IO**n, system develops a candidate set of migratable VMs, V^s, that can reduce applications' overall execution time. Then, the proposed VM migration policy employs a VM mapping procedure to select an appropriate cloudlet for provisioning VMs.

A. Selecting Candidate Set of VMs for Migration

For each provisioned VM $u \in V$, our proposed PRIMIO system calculates an urgency factor U_u that determines the criticality of migrating VM. Before calculating the urgency factor, we have to determine the predicted candidate cloudlet set,$V^{c}_{j,u}$ for the VM u, from which users can get their services. There are several algorithms in the literature for predicting the user location in the upcoming VM schedule; in this work, we use ENDA [30]. Recall that the key notion of a VM migration is to minimize the service execution time and hence to meet the service deadline of the hosted user application. Therefore, we calculate the urgency factor as follows,

$$U_u = T^{p,r}_{u} \times \gamma^{u}_{p,r} + T^{p,d}_{u} \times \gamma^{u}_{p,d}, \tag{10}$$

here, the urgency factor is a linear combination of two attributes of a task: first is the task-execution time required penalty $T^{p,r}_{u}$ and second is the task-execution time deadline penalty $T^{p,d}_{u}$ for a VM u. $T^{p,r}_{u}$ is the difference between the task-execution time, which is required to execute in the currently provisioned VM, and the required task-execution time. In the same way, $T^{p,d}_{u}$ can be defined as the difference between the task-execution time and the task-execution deadline time. These two task-execution time penalties can be calculated in the following way,

$$T^{p,r}_{u} = T^{e}_{u} - T^{r}_{u} \tag{11}$$

$$T^{p,d}_{u} = T^{e}_{u} - T^{d}_{u} \tag{12}$$

$$T^{r}_{u} = \frac{\sum_{i \in A^{r}_{u}, t=u(t)} T^{i,t}_{r}}{|A_{u,r}|} \tag{13}$$

where,$A_{u,r}$ is the set of previous task execution time required by the VM u to execute task type t. T^{e}_{u},T^{d}_{u} and T^{r}_{u} are the task-execution time required to execute the task in the provisioned VM, task-execution time deadline and required task execution time, respectively. T^{d}_{u} is set by user, where T^{r}_{u} is defined by the system itself. In Eq.(10), $\gamma^{u}_{p,r}$ and $\gamma^{u}_{p,d}$ are task-execution time penalty coefficients. These coefficients defined how much weight we have to set to each task execution time penalties based on the system and the user preferences. $\gamma^{u}_{p,r}$ and $\gamma^{u}_{p,d}$ can be defined as,

$$\gamma^{u}_{p,r} = \begin{cases} 1, & T^{r}_{u} \leq T^{e}_{u} \\ T^{p,r}_{u} \times \beta^{u}_{d,r}, & \text{otherwise} \end{cases} \tag{14}$$

$$\gamma^{u}_{p,d} = \begin{cases} 0, & T^{e}_{u} \leq T^{d}_{u} \\ T^{p,d}_{u} \times (1 - \beta^{u}_{d,r}), & \text{otherwise} \end{cases} \tag{15}$$

here $\beta^{u}_{d,r}$ is the user application preference. With $\gamma^{u}_{p,r}$ and $\gamma^{u}_{p,d}$, $\beta^{u}_{d,r}$ helps the system to determine when the task must be migrate as well as gives the system a degree of flexibility to migrate a VM from the cloudlet and provisioned in a new cloudlet. $\beta^{u}_{d,r}$ can defined as,

$$\beta^{u}_{d,r} = \begin{cases} 1, & \text{if } d \text{ attribute is surely preferred than } r \\ (0.5, 1), & \text{if } d \text{ attribute is partially preferred than } r \\ 0.5, & \text{if no preference} \\ (0, 0.5), & \text{if } r \text{ attribute is partially preferred than } d \\ 0, & \text{if } r \text{ attribute is surely preferred than } d. \end{cases} \tag{16}$$

After that, VMs are sorted according to urgency factor in decreasing order. According to this order, candidate-VM selection algorithm chooses a set of VMs based on the available resources in cloudlets accessible to each VMs. Nevertheless, during this selection the VM migration system does not map a VM into a cloudlet, but rather just selects a set of provisioned VMs from the cloudlet which need to be migrated. If the VM type or the required resources of a VM is ful-filled by any accessible VM, then it will be selected; otherwise, it must wait for the next scheduling interval. Therefore, the total VM resources are required by all the selected VMs must be less than or equal to the VM resources available in the accessible cloulets. The prioritized VM selection method is summarized in Algorithm (1).

Algorithm 1 Candidate VM Set Selection Algorithm, at cloudlet

```
    INPUT: VM set $V$ and accessible cloudlet set $C_u$ for each VM $u$.
    OUTPUT: Candidates VM list for migration.
 1: Calculate urgency priority $U_u, \forall u \in V$
 2: Reorder the urgency priority $U_u, \forall u \in V$ in following way,
    $U_1 \geq U_2 \geq U_3 \geq ....U_N$
 3: $V^s = \emptyset$
 4: repeat
    for each user VM $u \in V$
 5:     for (for each VM $v \in V^c_{j,u}$) do
 6:         if $(u^r \geq v^r | r \in R^u_i)$ then
 7:             $V_{v^c} = V_{v^c} \setminus \{v\}$
 8:             $V^s = V^s \cup u$
 9:         end if
10:     end for
11: until all VMs are provisioned or resources are utilized
```

The proposed VM migration system requires each task be associated with two task execution deadlines: (1) task-execution deadline time, T^d_u, which is set by the user application; and (2) the required task-execution time, T^r_u, which is defined by the system itself. The task execution time, T^e_u, can be greater than the T^r_u; however, it must be less than or equal to T^d_u. These two task-execution times give the VM migration system flexibility in initiating VM migration. T^r_u, especially, helps the system to initiate the VM migration in order to minimize the service downtime. For instance, if T^e_u exceeds T^r_u, the system may initiate VM migration for that particular VM if enough resources are available. However, if the T^e_u in a VM does not exceed the T^d_u but does exceed T^r_u and if available resources are limited, then the VM migration system gives more priority to those VMs which have higher need to be migrated to another cloudlet. Consequently, this approach has advantages for both the system and the user application, especially, when a small amount of time is needed to complete execution while available resources are limited.

B. Meta-heuristic Ant Colony-based VM migration algorithm

In our proposed VM migration approach, we jointly migrate a set of VMs to a set of cloudlets. That is, a cloudlet remaps a set of VMs to a set of cloudlets. This is a bin-packing problem, where a set of VMs is represented as bins while the cloudlets represent packs. These VMs must be packed with minimum number of cloudlets such that the total execution time of tasks is minimized; also, we have to minimize resouces wasted during packing. Thus, it becomes an NP-hard problem [44], i.e., no algorithm can provide a guaranteed optimal solution in polynomial time. For this reason, we have proposed a meta-heuristic Ant Colony based VM Migration algorithm to solve the VM migration problem. We use this meta-heuristic algorithm since it is problem-independent and does not require the benefit of any specificity of the proposed problem.

As swarm-optimization models are well-suited to building optimal solutions in very dynamic environments [15], we employ Ant Colony Optimization (ACO) to incorporate the dynamically changing cloud computing environment in the selection of optimal VM-cloudlet pair. Also, in a distributed environment, ACO helps to speedup computation, making it more suitable for the VM migration decision process. In addition, each cloudlet makes VM migration decisions individually by exploiting the information collected from neighboring cloudlets, mobile devices, and the master cloud. Furthermore, the ACO's learning capability of good solution features leads to the optimal solution and greatly decreases the selection probability of poor solutions [32].

The ACO is a meta-heuristic algorithm that uses the behavior of virtual ants to build a heuristic solution. A set of ants are created, with each ant trying to build a solution by solving sub-problems using heuristic information. The ants try to improve their solutions by exchanging information via pheromones. Each ant uses this pheromone trail and the local heuristic information to build a local optimal solution. Finally, all ants combine their local best solutions to infer an optimal solution.

Algorithm 2 Ant Colony Based VM Migration, at cloudlet i

```
    INPUT: VM set V and accessible cloudlet set $C_u$ for each VM $u$. System Parameters
    OUTPUT: VM-cloudlet pair.
 1: Initialize system tunning parameter $\alpha$
 2: Initialize system migration parameters $\gamma_l$ , $\gamma_g$
 3: Determine Candidate VM set $V^s$ using Algorithm 1
 4: Initialize ants set $A$
 5: Generate initial solution using FFVM Algorithm 3
 6: Calculate initial pheromone $\gamma_0$
 7: Set maximum iteration MAX_IT
 8: while ( do iteration $\leq$ MAX_IT)
 9:     for (Ant a $\in$ A) do
10:         k = 0
11:         repeat
12:             Select a VM v in cloudlet $j$ for VM $u_k \in V^s$ using Eq. 22
13:             k = k+1
14:         until every VM is provisioned to a cloudlet
15:         for (VM k $\in V^s$) do
16:             Update the local pheromone using Eq. 24
17:         end for
18:     end for
19:     Update the global pheromone using Eq. 25
20:     iteration = iteration+1
21: end while
22: Return VM-cloutlet pairs
```

1) Pheromones and Initial Pheromone Calculation: In ACO algorithm, pheromones represent the desirability of choosing a solution. In our proposed $PRIMIO$ algorithm, the pheromones represent the desirability of assigning a VM to a cloudlet. Each ant starts with an initial pheromone value for each VM to cloudlet pair. The initial solution is generated using a greedy First Fit(FF) VM migration approach, which is listed in Algorithm 3.

Algorithm 3 First Fit VM Migration, at cloudlet i

```
    INPUT: VM set V and accessible cloudlet set $C_u$ for each VM $u$.
    OUTPUT: VM-cloudlet pair in initial solution.
1: for (VM u ∈ V) do
2:     for (VM v in $V^c_{j,u}$) do
3:         if ($T_{u,v} \leq T^D_u$ And $\forall_{r \in R^u_i} R^{D,r}_u \leq R^r_v$ ) then
4:             Assign VM $i$ to cloudlet $k$
5:             Break
6:         end if
7:     end for
8: end for
```

The initial pheromone value is calculated by summing the total task-execution times and the resources over-provisioned in each cloudlet and taking the inverse of the amount. Initial pheromone value for each ant is calculated as follows,

$$\tau_0 = \sum_{u \in V^s_i} \sum_{v \in V^c_{j,u}} \frac{1}{T_{u,v} + \sum_{r \in R^u_i} R^{o,r}_{u,v}} \times y_{u,v} \tag{17}$$

where, $y_{u,v}$ is a binary variable, which is defined as,

$$y_{u,v} = \begin{cases} 1, & \text{if } (u,v) \in S_0 \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

2) Calculation of Heuristic Value: To build a solution each ant uses the local heuristic value to select a cloudlet for a VM. This heuristic value defines the favorability of choosing a cloudlet for a VM to construct the solution. As $PRIMIO$ tries to jointly optimize the objectives of total task-execution time and resource over-provisioning in cloudlets. So, the local heuristic value has to be defined in such a way that the system can optimize the task execution time and resource over-provisioning to reflect the system objectives. Local heuristic is defined as,

$$\eta_{u,v} = \alpha \times \eta^E_{u,v} + (1-\alpha)\, \eta^R_{u,v} \tag{19}$$

where, α is the system parameter defines the relative weight between the objectives of total task execution time and resource over-provisioning of cloutlets. It can be tuned according to the system environment. $\eta^E_{i,j}$ and $\eta^R_{i,j}$ is the objectives of total task execution time and the resource over-provisioning respectively, if VM migrates from cloudlet i to cloudlet j. $\eta^E_{i,j}$ and $\eta^R_{i,j}$ are defined in Eq. (20) and Eq. (21), respectively.

$$\eta^E_{u,v} = \sum_{k=1}^{u} \sum_{v \in V^u_{j,u}} \frac{1}{T_{k,v}} \tag{20}$$

$$\eta^R_{u,v} = \sum_{r \in R^u_i} \sum_{k=1}^{u} \sum_{v \in V^c_{j,u}} \left(1 - R^{o,r}_{k,v}\right) \tag{21}$$

3) Cloudlet Selection: Pseudo-random Proportional Rule: While constructing the solution, each ant selects a cloudlet for each VM using a pseudo-random proportional rule, which can be defined as follows,

$$v = \begin{cases} \arg\max_{k \in V^c_{j,u}} \left([\tau_{u,k}]^\alpha \times [\eta_{u,k}]^\beta \right), & \text{if } q \leq q_0 \\ s, & \text{otherwise} \end{cases} \tag{22}$$

where, q is a random numbers uniformly distribute in [0,1] and q_0 is the system parameter on the range [0,1]. $\tau_{i,k}$ is the pheromone value of selecting cloudlet k for VM i. When $q \leq q_0$, (i.e., exploitation) ant selects a cloudlet j for a VM i, where the multiplication of $[\tau_{i,k}]^\alpha \times [\eta_{i,k}]^\beta$ gives the highest value among the all possible choice of cloudlets. Here, α and β are the system parameters, denoting the relative importance of pheromone value and the local heuristic value for choosing a cloudlet j for a VM in the cloudlet i. If the random number q is greater than q_0, i.e. in case of exploration, an ant z choose a cloudlet j using the probability $p^z_{i,j}$, which can be defined as,

$$p^z_{u,v} = \begin{cases} \frac{\left([\tau_{u,k}]^\alpha \times [\eta_{u,k}]^\beta\right)}{\sum_{k \in V^c_{j,u}} \left([\tau_{u,k}]^\alpha \times [\eta_{u,k}]^\beta\right)}, & \text{if } q \leq q_0 \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

4) Local Pheromone Update: When an ant chooses a VM pair to construct a solution, it immediately updates the local pheromone value in relation to the initial pheromone value. Local pheromones are updated by each ant using the following relation,

$$\tau_{u,v}(t+1) = (1-\gamma_l) \times \tau_{u,v}(t) + \gamma_l \tau_0 \tag{24}$$

where, γ_l is the system parameters, denoting the relative importance of current pheromone value at time t, $\tau u, v(t)$.

5) Global Pheromone Update: Global pheromone values are updated for each pair of VM and cloudlet only when all ants constructed their local optimal solutions and updated the global optimal solution. The global pheromone values are updated using the following relation,

$$\tau_{u,v}(t+1) = (1-\gamma_g) \times \tau_{u,v}(t) + \gamma_g \Delta\tau_{u,v} \tag{25}$$

where the γ_g is the global pheromone system parameter, this parameters can be tuned according to the system objectives. $\Delta\tau_{u,v}$ is the global pheromone value for the updated global solution, which is defined as

$$\Delta\tau_{u,v} = \begin{cases} \tau_{u,v}, & \text{if } (u,v) \in PGS \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

where, PGS is the global solution set of the selected VM u for migration and the target VM v where the migration will be occurred.

6) VM Migration Initiator: Ant colony optimization optimally selects a cloudlet for a VM. After each VM is mapped to a particular cloudlet, the VM migration initiator is invoked, which determines whether the PRIMIO algorithm selects the current cloudlet for the VM as a target cloudlet. If the target and current cloudlets for a VM are different then the VM migration manager initiates a VM migration event. Otherwise, the VM is provisioned to the current cloudlet and no VM migration event initiation is required.

VI. Performance Evaluation

In this section, we assess the performance of our proposed PRIMIO method through test-bed experimentation. We compare the performance of proposed PRIMIO method with 'No Migration,' 'Task-centric migration' [14], GAVMM [24] and mobility-based 'greedy migration' methods. In implementing the 'No Migration' method, VMs are not migrated even though a cloudlet is overloaded or the user moves to cloudlets; the cloudlet, where the VM is provisioned, executes the task and forwards the result to the destination cloudlet. In implementing 'Task-centric migration,' an individual task is migrated to another cloudlet, following the task's mobility pattern and cloudlet's computational load, in order to improve task execution time. 'GAVMM', employs a genetic algorithm to select a target cloudlet based on the user mobility and cloudlet sever load. On the other hand, *Greedy VM Migration* method migrates a VM to a cloudlet from which the user is receiving the highest WiFi signal strength, without considering the load of that cloudlet initiating VM migration.

A. Experimental Environment

As test bed to evaluate performance of PRIMIO, we used ten cloudlets, each with the following computational resources [1.5-3.0]GHz processor, [8-16]GB memory and [250-350GB] SATA hard disk. All cloudlets were interconnected with [2-20]Mbps Ethernet links. Users access the cloudlets using mobile devices with varying computation capacity. Each user device is connected with cloudlet APs through a WiFi IEEE 802.11g interface. In this experiment, we used two Galaxy Grand 2, two Sony Xperia Z, and six Symphony Android location-enabled devices.

Tasks in the mobile devices are generated by following a Poisson distribution. The tasks are text, a document, or an image which is uploaded to cloudlet for analysis. After that, the result will be pushed back to the user's mobile device. Here, α is a system parameter, the value of which is determined based on the user task. In our experiment, we set the system parameter to 0.65 through rigorous simulation and also because the goal of task is to reduce the task-completion time. Based on rigorous numerical simulation and task type, we set other experiment parameters ω_0, γ_l and γ_g. The parameters used in testbed experiments are enlisted in Table VI-A.

TABLE II
Simulation Parameters

Parameters	Value
Total number of Ant	20
Maximum iteration MAX_IT	1200
Mobility Speed	$[4.5 \sim 7.0]$ Km/h
Task Lifetime	$[4 \sim 14]$ minutes
Data Footprints	$[0.5 \sim 10] \times 10^6$ bytes
α	0.65
γ_l	0.30
γ_g	0.40
Simulation Time	20 minutes

B. Performance Metrics

To evaluate the performance of the proposed PRIMIO approach to VM migration compared to state-of-the-art VM migration methodologies, we consider the following metrics:

- *Average Task-Completion Time:* calculated by averaging task execution time for a set of tasks executing in the cloudlets using different VM migration mechanisms. A VM migration method with decreasing average task-completion time increases the quality of user experience (QoE) by executing the task in the cloudlets rather than executing in the mobile devices.
- *Resources Over-Provisioned (%):* measured by calculating the amount of extra resources are provisioned compared to the resources required. We measure the percentage of over-provisioned resources using Eq. 9. If the percentage of over-provisioned resources increases for a VM migration method, that degrades the overall system performance.
- *Successful Task Completion (%):* calculated as the percentage of tasks, which are provisioned to cloudlets and which successfully completed their executions within the predefined task execution deadline. This metric helps to evaluate the probability that a VM migration approach will finish the execution of a task within deadline.
- *VM Migration Overhead:* is the ratio of the total number of migrations are required to execute a set of tasks and the total number of tasks finishing execution within the predefined deadline. This metric indicates the performance of VM migration method in terms of utilization of provisioned resources.

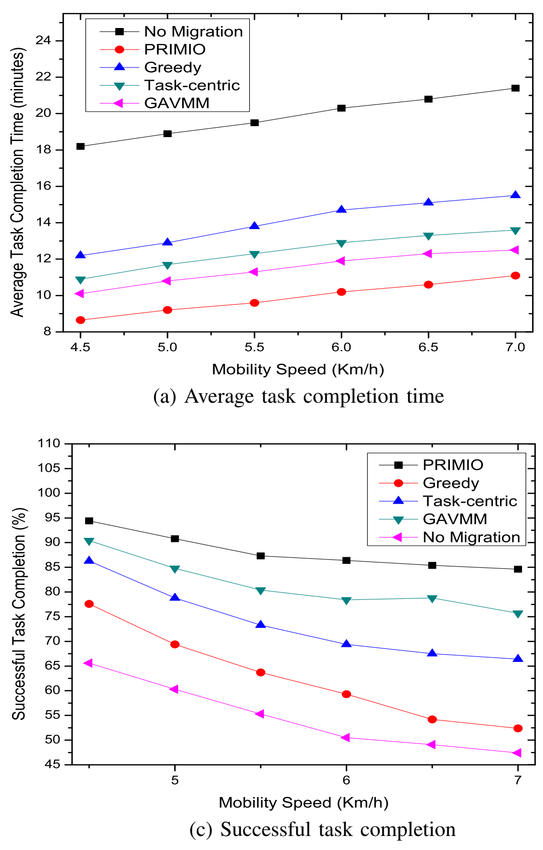
C. Experimental Results

We studied the performance of our proposed $PRIMIO$ method by varying user mobility, task lifetime, and data footprint. In simulation, we only considered a pedestrian walking model.

1) Impacts of user mobility: We gauge the impact of user mobility on average task-execution lifetime in cloudlets, over-provisioned resources, task-executions completed within deadline and the VM migration overhead. To study the impact of user mobility, we vary user speed from $[4.5 \sim 7.0]$Km/hr, which follows the pedestrian walking model.

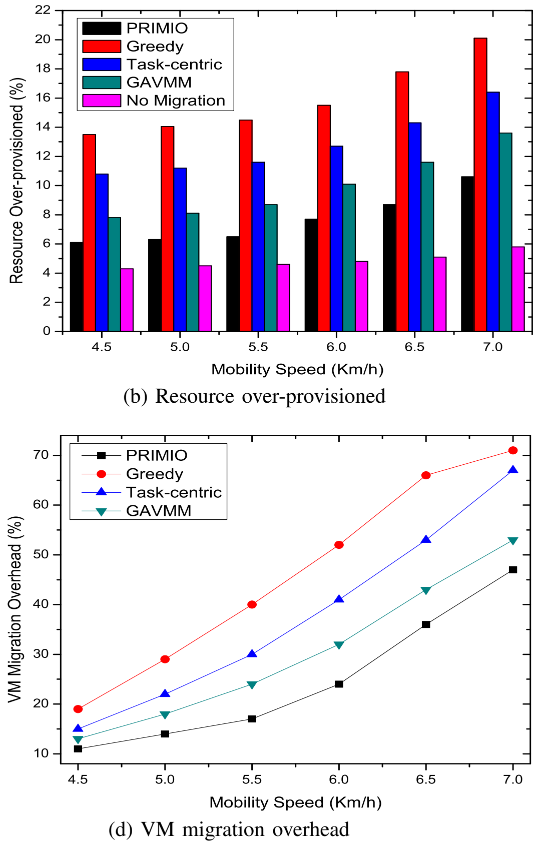
Fig. 3 (a) shows that the average task lifetime increases with increased mobility speed, because it causes frequent migration of VMs, increasing the service downtime. Moreover, the communication latency between the user and the cloudlet is also increased. Hence, the graph indicates that No Migration policy experiences higher average task-execution time. However, our proposed VM migration policy, $PRIMIO$, reduces the average task-execution lifetime as it migrates VMs based not only on the user mobility pattern but also it considers computational load of cloudlets. On the otherhand, other VM Migration approaches do not effectively utilize the user mobility to initiate the VM migration decision.

Fig. 3 (b) depicts that the percentage of over-provisioned resources increases with the higher mobility speeds. As user



(a) Average task completion time

(c) Successful task completion



(b) Resource over-provisioned

(d) VM migration overhead

Fig. 3. Impacts of mobility speed

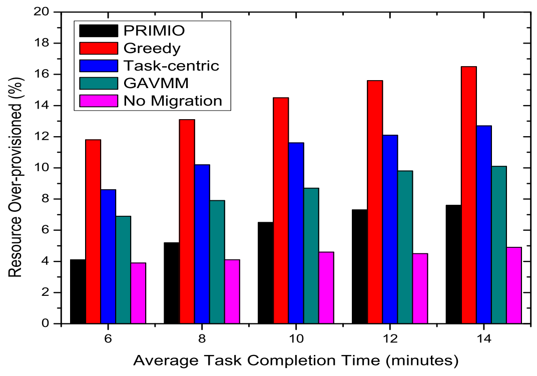
mobility speed increases, the tasks in the cloudlets also migrate more frequently, thus migrating to VMs with greater capacity than the required. The No VM migration approach, for example, does not increase the percentage of resource over-provision, as it does not migrate any VM. Even though, all the VM migration approaches increase the resource over-provision, our proposed $PRIMIO$ approach does not significantly increase the over-provisioned resources, because it considers a joint VM-migration approach. This helps the cloudlet system to minimize over-provisioned resources across the system as a whole. By contrast, the task-based and greedy VM migration approaches utilize a single VM migration approach, making migration decisions without considering the other provisioned VMs. Therefore, these VM migration approaches increase the over-provisioned resource as user mobility increases.

Fig. 3 (c) shows that the percentage of successful task execution within deadline degrades with increased user mobility speed. One possible explanation is that increased user mobility also increases communication delay with the VM provisioned in a cloudlet and increase the task lifetime which in turns miss the task execution deadline. For this reason, the No VM migration approach has a lower percentage of tasks executed within deadline. The $PRIMIO$ VM migration approach migrates the VM to a cloudlet by following user mobility and cloudlet load. However, greedy and task-centric VM migration approaches can not effectively utilize user mobility. Moreover, the greedy VM migration approach does not consider the load of the provisioned cloudlet.

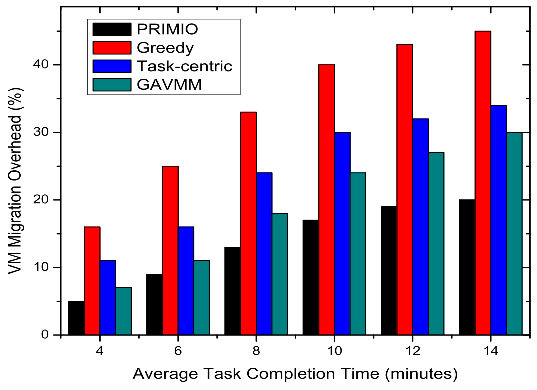
Fig. 3 (d) depicts that increased user mobility speed also increases VM migration overhead. We do not study the No VM migration approach here, as it migrates no VM and thus incurs no migration overhead. Although, our proposed VM migration approach $PRIMIO$ increases VM migration overhead with the increased user mobility speed, it does not increase significantly compared to others. This is for two main reasons. First, $PRIMIO$ uses user mobility information in initiating VM migration decisions, thus decreasing task execution lifetime and increases the percentage of successful completed of task execution within the execution deadline. Secondly, PRIMIO considers cloudlet load in the VM migration decision process, which effectively reduces the total number of VM migrations across the whole task lifetime. Even though greedy and task-centric VM migration approaches consider the user mobility and thus reduce the total number of VM migration, neither could significantly increase the percentage of successful task execution within deadline, therefore failing to reduce VM migration overhead.

2) Impacts of Task Lifetime: We measure the impacts of task lifetime on the percentage of over-provisioned resources and VM migration overhead by varying the task lifetime on the cloudlets from [6 $\sim$ 14] minutes. During this simulation, we set user mobility speed to approximately 5.5 Km/h.

Fig. 4 (a) indicates that the percentage of over-provisioned resources increases with average task lifetime. Nevertheless, the over-provisioned resources do not increase significantly



(a) Resource over-provisioned



(b) VM migration overhead

Fig. 4. Impacts of average task completion time

in $PRIMIO$, because, it employs a joint VM migration approach that effectively utilizes the system resources. On the other hand, the Greedy VM migration approach increases over-provisioned resources more rapidly as it only considers the user mobility without looking at cloudlet resources. Meanwhile, the task-centric VM migration method takes a single VM migration approach, which fails to effectively utilize cloudlet resources. Finally, the No VM migration approach, eventhough it does not increase the over-provisioned resources, but it negatively impacts on other aspects.

Fig. 4 (b) depicts that increased task execution lifetime in the cloudlets also increases VM migration overhead. Obviously, a task that executes for a longer period of time would incur more VM migration overhead. However, if migration increases rapidly with regard to task lifetime, it degrades both system performance and user application's quality of service. This graph indicates that, compared to our proposed $PRIMIO$ VM migration approach, other methods more greatly increase VM migration overhead. Moreover, VM migration overhead in the $PRIMIO$ approach reaches a steady state when task execution lifetime is close to 10 minutes. Because, $PRIMIO$ utilizes both the joint-VM migration approach and user mobility to initiate VM migration decisions.

3) Impacts of Data Footprints: We studied impacts of data footprint, which is the data associated with the VM, on average task lifetime, successful task execution within deadline and VM migration overhead of different VM migration approaches. During this study, we set the user mobility speed to approximately 5.5 Km/h. We varied the data footprint size from $[0.5 \sim 10] \times 10^6$.

Fig. 5 (a) shows that average task execution lifetime increase as data footprint increases. Nonetheless, compared to the other VM migration approaches, $PRIMIO$ does not significantly increase the average task lifetime, because it considers a data-transfer penalty in the VM migration decision process. Moreover, in the graph we can see that the average task lifetime reaches a steady state when the data footprint size is greater than approximately 4×10^6. Larger than this footprint, the $PRIMIO$ method refrains from transferring a VM closer to the user in order to reduce the total task execution time. On the other hand, other VM migration approaches only consider user mobility when imitating VM migration and, thus they increase the average task lifetime.

Fig. 5 (b) depicts that increased data footprint invresely affects the percentage of task completion within the deadline. However, the task completion percentage does not decrease in our proposed $PRIMIO$ compared to the other approaches. As previously stated $PRIMIO$ considers the data transfer time before initiating any VM migration decision. On the other hand, the percentage of task completion in the No VM migration approach decreases significantly, as users move from one access point to another, the communication delay increases, and thus it increases transfer time of final task result. Therefore, No VM migration approach significantly reduces the percentage of tasks completed within deadline.

Fig. 6 shows that increased data footprints also increase VM migration overhead. The greedy and task-centric VM migration policies that do not effectively consider migration time, increase not only total number of VM migrations but also decrease number of tasks completed within deadline. Nonetheless, our proposed method $PRIMIO$ does not increase the VM migration overhead compared to other polices, mainly because $PRIMIO$ decreases the total number of VM migrations as the data footprints size increases and also increases the total number of tasks completed within deadline. Thus, the proposed algorithm demonstrated its superiority over existing algorithms.

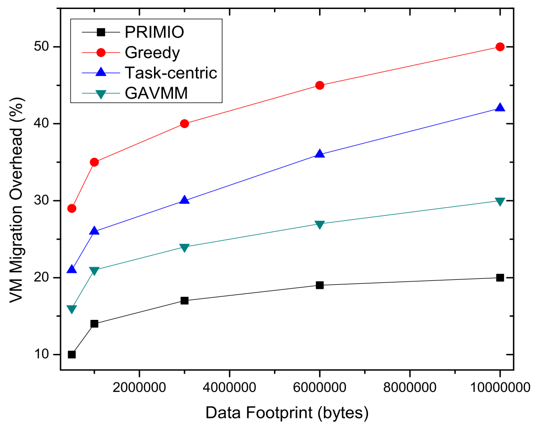
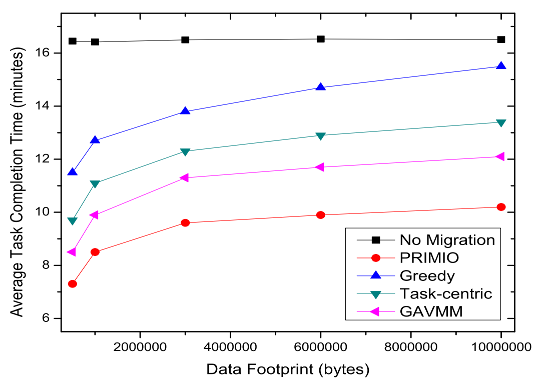
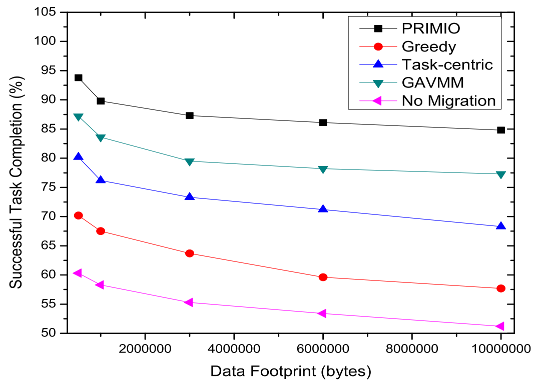


Fig. 6. Impacts of data footprints: VM migration overhead



(a) Average task completion time



(b) Successful task completion

Fig. 5. Impacts of data footprints

VII. Conclusion and Future Work

In this paper, we have proposed a mobility- and resource-aware joint virtual-machine migration model for heterogeneous mobile cloud computing systems to improve the performance of mobile Smart health care applications in a Smart City environment. Here, we address research challenges to reduce task-completion times as well as to reduce resource over-provisioning in mobile cloud computing that executes both computationally and Big Data-intensive healthcare tasks. The proposed PRIMIO model initiates VM migration by considering user mobility and computational load of a cloudlet. As PRIMIO exploits users' mobility in achieving an optimal solution, it effectively brings cloud resources closer to the user. At the same time, PRIMIO considers the load of the cloudlet to which the system wants to migrate the VM, thereby reducing total number of migrations across the entire task-execution time. Furthermore, we considered rates of resource over-provisioning during VM migration, allowing the overall system to utilize computing resources optimally. Left for our future work is how to further optimize the task-computation time and data-access latency through considering the presence of fog clouds and crowd sourcing. In this aspect, we will explore the emerging edge computing technology to further optimize the task execution time while considering mobility and context-awareness.

References

[1] J. H. Abawajy and M. M. Hassan, "Federated internet of things and cloud computing pervasive patient health monitoring system," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 48–53, 2017.

[2] E. Ahmed, M. Imran, M. Guizani, A. Rayes, J. Lloret, G. Han, and W. Guibene, "Enabling mobile and wireless technologies for smart cities," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 74–75, 2017.

[3] M. Basiri, A. Z. Azim, and M. Farrokhi, "Smart city solution for sustainable urban development," *European Journal of Sustainable Development*, vol. 6, no. 1, pp. 71–84, 2017.

[4] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on*, May 2007, pp. 119–128.

[5] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities," in *High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on*. Ieee, 2008, pp. 5–13.

[6] 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*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 301–314.

[7] R. Cimler, J. Matyska, and V. Sobeslav, "Cloud based solution for mobile healthcare application," in *Proceedings of the 18th International Database Engineering & Applications Symposium*, ser. IDEAS '14. New York, NY, USA: ACM, 2014, pp. 298–301.

[8] E. Cuervo, A. Balasubramanian, D.-k. 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*, ser. MobiSys '10. New York, NY, USA: ACM, 2010, pp. 49–62.

[9] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[10] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[11] C. Doukas, T. Pliakas, and I. Maglogiannis, "Mobile healthcare information management utilizing cloud computing and android os," in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, Aug 2010, pp. 1037–1040.

[12] E.-M. Fong and W.-Y. Chung, "Mobile cloud-computing-based healthcare service by noncontact ecg monitoring," *Sensors*, vol. 13, no. 12, pp. 16 451–16 473, 2013.

[13] L. Gkatzikis and I. Koutsopoulos, "Migrate or not? exploiting dynamic task migration in mobile cloud computing systems," *Wireless Communications, IEEE*, vol. 20, no. 3, pp. 24–32, June 2013.

[14] L. Gkatzikis and I. Koutsopoulos, "Mobiles on cloud nine: efficient task migration policies for cloud computing systems," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 204–210.

[15] M. Guntsch, M. Middendorf, and H. Schmeck, "An ant colony optimization approach to dynamic tsp," in *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO'01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 860–867.

[16] M. M. Hassan, "Cost-effective resource provisioning for multimedia cloud-based e-health systems," *Multimedia Tools and Applications*, vol. 74, no. 14, pp. 5225–5241, 2015.

[17] M. M. Hassan, H. S. Albakr, and H. Al-Dossari, "A cloud-assisted internet of things framework for pervasive healthcare in smart city environment," in *Proceedings of the 1st International Workshop on Emerging Multimedia Applications and Services for Smart Cities*. ACM, 2014, pp. 9–13.

[18] M. M. Hassan, K. Lin, X. Yue, and J. Wan, "A multimedia healthcare data sharing approach through cloud-based body area network," *Future Generation Computer Systems*, vol. 66, pp. 48–58, 2017.

[19] M.S. Hossain, M. Moniruzzaman, G. Muhammad, A. Ghoneim and A. Alamri, "Big data-driven service composition using parallel clustered particle swarm optimization in mobile environment," *IEEE Trans. Service Computing*, vol. 9, no. 5, pp. 806-817, 2016.

[20] M.S. Hossain, S.A. Hossain, A. Alamri, and M.A. Hossain, "Ant-based

service selection framework for a smart home monitoring environment," *Multimedia tools and applications*, vol. 67, no. 2, pp. 433-453, 2013.

[21] M. S. Hossain, "Patient state recognition system for healthcare using speech and facial expressions," *Journal of medical systems*, vol. 40, no. 12, p. 272, 2016.

[22] M. S. Hossain and G. Muhammad, "Cloud-assisted industrial internet of things (iiot)–enabled framework for health monitoring," *Computer Networks*, vol. 101, pp. 192–202, 2016.

[23] M. S. Hossain and G. Muhammad, "Healthcare big data voice pathology assessment framework," *IEEE Access*, vol. 4, pp. 7806–7815, 2016.

[24] M. Islam, A. Razzaque, and J. Islam, "A genetic algorithm for virtual machine migration in heterogeneous mobile cloud computing," in *2016 International Conference on Networking Systems and Security (NSysS)*, Jan 2016, pp. 1–6.

[25] A. Khan, M. Othman, S. Madani, and S. Khan, "A survey of mobile cloud computing application models," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 393–413, First 2014.

[26] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 945–953.

[27] P. Kulkarni and T. Farnham, "Smart city wireless connectivity considerations and cost analysis: Lessons learnt from smart water case studies," *IEEE Access*, vol. 4, pp. 660–672, 2016.

[28] R. Kumari, S. Kaushal *et al.*, "Application offloading using data aggregation in mobile cloud computing environment," in *Leadership, Innovation and Entrepreneurship as Driving Forces of the Global Economy*. Springer, 2017, pp. 17–29.

[29] P. G. J. Leelipushpam and J. Sharmila, "Live vm migration techniques in cloud environment : A survey," in *Information Communication Technologies (ICT), 2013 IEEE Conference on*, April 2013, pp. 408–413.

[30] J. Li, K. Bu, X. Liu, and B. Xiao, "Enda: Embracing network inconsistency for dynamic application offloading in mobile cloud computing," in *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, ser. MCC '13. New York, NY, USA: ACM, 2013, pp. 39–44.

[31] J. Liu, Y. Li, D. Jin, L. Su, and L. Zeng, "Traffic aware cross-site virtual machine migration in future mobile cloud computing," *Mobile Networks and Applications*, vol. 20, no. 1, pp. 62–71, 2015.

[32] J. Montgomery, M. Randall, and T. Hendtlass, "Structural advantages for ant colony optimisation inherent in permutation scheduling problems," in *Proceedings of the 18th International Conference on Innovations in Applied Artificial Intelligence*, ser. IEA/AIE'2005. London, UK, UK: Springer-Verlag, 2005, pp. 218–228.

[33] Z. L. Phyo and T. Thein, "Correlation based vms placement resource provision," *International Journal of Computer Science & Information Technology*, vol. 5, no. 1, p. 95, 2013.

[34] M. Rahimi, J. Ren, C. Liu, A. Vasilakos, and N. Venkatasubramanian, "Mobile cloud computing: A survey, state of art and future directions," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 133–143, 2014.

[35] C. C. Sasan Adibi, Nilmini Wickramasinghe, "Ccmh: The cloud computing paradigm for mobile health (mhealth)," *International Journal of Soft Computing and Software Engineering [JSCSE]*, pp. 403–410, 2013, 10.7321/jscse.v3.n3.61.

[36] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[37] M. Sneps-Sneppe and D. Namiot, "On mobile cloud for smart city applications," *arXiv preprint arXiv:1605.02886*, 2016.

[38] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *Global Communications Conference (GLOBECOM), 2013 IEEE*, Dec 2013, pp. 1291–1296.

[39] H. N. Van, F. Tran, and J.-M. Menaud, "Sla-aware virtual resource management for cloud infrastructures," in *Computer and Information Technology, 2009. CIT '09. Ninth IEEE International Conference on*, vol. 1, Oct 2009, pp. 357–362.

[40] U. Varshney, *Pervasive Computing and Healthcare*. Boston, MA: Springer US, 2009, pp. 39–62.

[41] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou, and Z. Liu, "Joint virtual machine assignment and traffic engineering for green data center networks," *SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 3, pp. 107–112, Jan. 2014.

[42] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Military Communications Conference (MILCOM), 2014 IEEE*, Oct 2014, pp. 835–840.

[43] I. Yaqoob, I. A. T. Hashem, Y. Mehmood, A. Gani, S. Mokhtar, and S. Guizani, "Enabling communication technologies for smart cities," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 112–120, 2017.

[44] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.