International Journal of Computer Applications (0975 - 8887)
Volume 175 - No.6, October 2017

An Optimal Task Scheduling Mechanism for Mobile Cloud Computing

Md. Erfan
Department of Computer Science and Engineering
University of Barisal

Farhana Huq
Department of Computer Science and Engineering
Northern University, Bangladesh

Md. Shariful Islam
Institute of Information Technology
University of Dhaka

ABSTRACT

Mobile devices limited storage and computation capabilities are largely affected by the compute intensive, resource intensive or energy drain applications. These limitations of the mobile devices can be eliminated with the help of mobile cloud computing by delivering the energy drain or computing intensive parts of the task to more resourceful resources and receiving the result from the resources. This process (a.k.a code offloading) helps the mobile device to increase performance and reduce energy consumption. We have proposed an optimal task scheduling code offloading mechanism which optimally identify the remote executable tasks and also identify the remote VM to execute the task. We have also proposed some greedy algorithms to evaluate the result of our proposed task scheduling algorithm. Herein, the local execution time, maximum allowable time and communication latency information and scheduling a task based on the remote VM execution time and queuing time etc. information are served to the master cloud. In our experimentation, we have used an image processing application to validate the proposed system. From our evaluation we show that tasks executed on high capacity VM improve the overall execution time comparing with local mobile device execution. It also shows that the proposed mechanism for offloading task from mobile device to remote resource perform efficiently for resource heterogeneity.

General Terms

Mobile device, offloading, scheduling.

Keywords

Mobile cloud computing, code offloading, migration, partitioning.

1. INTRODUCTION

The mobile user's density and number of mobile devices in the world are becoming high and mobile applications are gaining popularity in recent years. The increasing developments of that applications will revolutionize many sectors of our economy, including business, health care, social networks, environmental monitoring, and transportation, etc. Mobile devices (e.g., smartphone and tablet PC) have become an important part in today's life because of its desktop like functionality [1]. Nowadays, it is the most effective and efficient communication tools which is not bounded by time and place. Mobile applications executed locally or on remote servers connected via wireless network providing rich experiences of various services to mobile users. For the development of IT technology, commerce, e-commerce, and industry fields, mobile computing becomes a very powerful trend [2]. Having so many advantages, it faces so many challenges such as executing resource intensive applications like image and video processing, object or face recognition [3] decreases performance of the device. As it is equipped with accelerometer, digital compass, gyroscope, GPS, microphone, camera, and Wi-Fi, etc. requiring huge computation in background of the devices which increases the complexity [4]. The limiting factor of mobile devices in their resources (e.g., battery life, storage, and bandwidth) [5] and communications (e.g., mobility and security) [6, 7] that hinder the improvement of service quality. The limitations of the mobile devices can be mitigated by migrating either the entire application or parts of it to remote computing devices to gain benefits. Cloud computing takes these responsibility for an application by offering infrastructures, platforms and software at low cost. Cloud computing makes it possible for users to effectively utilize resources in an on-demand basis. In order to mitigate the limitations of mobile devices and mobile applications, Mobile Cloud Computing (MCC) [8, 9] is introduced as an integration of cloud computing into the mobile environment. MCC executes high computational or energy hungry applications to the cloud which allows the mobile users to use the cloud as Infrastructure as a Service (IAAS), Software as a Service (SAAS) and Platform as a Service (PAAS) at low cost. The ultimate goal of MCC is to enable execution of rich mobile applications on a huge number of mobile devices, with an increased user experience. As more computation are performed into the mobile devices, performances demanded by smartphones is increasing at a much faster rate than improvement in battery technology. To improve performance and battery lifetime, the huge computation required applications can be executed on mobile devices by migrating computational tasks to nearby servers using MCC known as computation offloading or code offloading. Traditionally, computations are offloaded to rich remote resources servers [10]. In code offloading, the entire application or part of it is migrated to remote server for performance improvement [11, 12], increasing the battery life [13] and reducing the cost of execution. In code offloading, applications can be partitioned as methods, classes or threads statically [14, 15] or dynamically [16, 17, 18, 19] considering variable parameters such as server workload, data rate, etc. This offloading to remote cloud often leads much overhead such as high latency and

low bandwidth as the real cloud provider may be located far away from the user.
Considering those issues mentioned above, the objective of this research is to optimally identify the remote virtual machine for offloadable task in mobile cloud computing which is a complex and challenging task. It is required to optimally identify the remote VM where the compute intensive task should be offloaded. The applications located in a mobile device have a maximum amount of time to execute, maximum allowable time are served to the master cloud profiler. Those information are used by the scheduler to optimally select the remote virtual machine.

2. RELATED WORK

This section describes a brief discussion on the existing code offloading mechanisms.In code offloading, tasks can be method, class, thread or data depending on the partitioning level. The state of the art offloading mechanisms can be classified into two categories named partitioning based mechanism and scheduling based offloading mechanism.

2.1 Task Partitioning Based Mechanisms

In code offloading, tasks can be method, class, thread or data and that tasks are delivered to the remote server in order to increase performance. The partitioned code are annotated to identify the local and remoteable task at runtime. The decision of partitioning a task can be static or dynamic. In static code partitioning, tasks are partitioned statically by the expert developer. On the other hand, dynamic partition of code partitioning the code dynamically considering variable parameters such as server workload, network connectivity, etc.

2.1.1 MAUI. Cuervo et al. [20] proposed a fine grained code offloading system for saving energy of mobile devices in making smartphones last longer with code offload called MAUI. MAUI introduces both static and dynamic decision of method level partitioning. Initially the developers annotated the remote executable methods as remoteable and locally executable methods as local. The proposed system automatically identified the remoteable and non-remoteable methods, and then automatically performed migration on application methods.

2.1.2 ThinkAir. Kosta et al. [21] proposed Thinkair that enables the parallelization of method execution using multiple Virtual Machine (VM) images for the enhancement of performance and battery lifetime of mobile devices. It introduces method level partitioning for code offloading and exploits the concept of smartphone virtualization in the cloud. The proposed system focuses on the elasticity and scalability of the cloud which increases the power of mobile cloud computing by parallelizing method execution using different VMs.The system is evaluated on the N-queen puzzle, face detection, virus scan, and image merger applications.

2.1.3 CloneCloud. Chun et al. [22] created CloneCloud which have enabled mobile application to migrate thread to device clones operating in a computational cloud. The system employs dynamic profiling and static analysis to partition the mobile application in order to optimize overall execution cost. Herein, the application is partitioned, send it to the clone, execute the code and re-integrating the migrated thread back to the mobile device. The selection of local and remote execution of methods is taken by the optimization solver. The purpose of the optimization solver is to deliver optimal application methods to the cloud from the mobile devices. The system has static analyzer for code offloading which require experienced domain experts. This paper consider limited environmental conditions and assume resources that are not available on the cloud. The CloneCloud system is evaluated on the virus scanning, image processing, and privacy preserving targeted advertising applications.
In task partitioning based mechanism, another research work Kemp et al. [23] proposed an offloading framework for Android named Cuckoo, including a runtime system, a resource manager application for mobile device user and a programming model for developers. Gordon et al. [24] propose code offload by migrating execution transparently called COMET, a multithreaded application that can be migrated freely between devices depending on the workload and use distributed shared memory for offloading. Flores et al. [25] proposed EMCO, where high computation required operations are identified and partitioned at code level and offloaded for remote processing. What computation to offload and how to structure the parallelism across the mobile devices and cloud, Ra et al. [26] proposed Odessa which dynamically makes offloading and data parallelism decisions for mobile interactive perception applications.

2.2 Task Scheduling Based Mechanisms

The previously discussed works consider only the mobile devices and the cloud for offloading code. But offloading code to distant remote cloud often leads too much overhead like low bandwidth and high latency as the actual cloud provider may be situated far away from the users. Therefore, researchers have considered existence of an intermediate server called master cloud attached to WiFi access point for code offloading. Since the resource allocation on the cloud is of immense importance due to the increasing quality of experience of the users, researchers are giving attention to address the issue of where to optimally offload a task for remote execution.

2.2.1 MAPCloud. MAPCloud [27] has proposed a two-tier cloud architecture with multiple quality of service factors like price, power and delay. To achieve a near optimal solution for allocating the task to a resource in the cloud, they proposed CRAM (Cloud Resource Allocation for Mobile Applications), a simulated annealing based heuristic. Herein, tier 1 consists of scalable and elastic public cloud like Amazon Web Services, Google Application Engine etc. This tier does not give the fine grain location granularity. That is why, the second tier consists of the local cloud where resources are connected to the access point. The fine grain location information of these resources is available. The middleware broker, located at the second tier of the architecture, manages and runs applications on this system. The broker has a registry of available resources and services at both tiers of the cloud. To determine whether the task is schedulable or not, the broker consults the registry for each application request which are considered as a workflow of task.

2.2.2 Online Algorithms for Location-Aware Task Offloading. Q. Xia et al. [28] have considered the user access limitation on the wireless network access points. The main objective of this solution is to determine the place of offloading the task with an aim to maximize the minimum residual energy ratio. Tier 1 includes the public cloud and tier 2 includes the local cloudlets. To select the server from the clouds, the authors have constructed a bipartite graph whose one set of vertices is the task at a time slot and the other set is the possible computing facility allocation. The edge between each node is the delay for executing the task in respective computing facility. The weight associated with each node is the

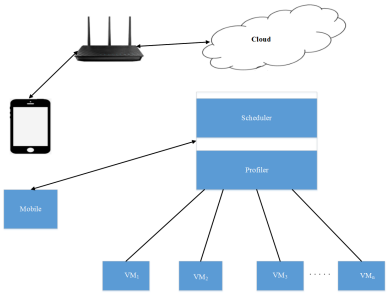


Fig. 1: High level design for the proposed system

energy cost calculated through the proposed energy cost model.

3. PROPOSED OPTIMAL TASK SCHEDULING MECHANISM

Different approaches are present in the current MCC literature for migrating task to the cloud from mobile devices. Most of these suffer from some drawbacks as depicted in the previous chapter. In this chapter, we describe our proposed system for task scheduling to the VM on cloud from the mobile devices. At first, we have depicted the architecture of our proposed system. The problem statement with an example scenario is narrated next. Finally, we formulate a mathematical model to select the VM from the cloud to execute the task.

3.1 System Architecture

The system follows the client-server architecture like several state-of-the-art cloud computing framework [3]. At mobile side, applications are running and selected offloadable methods are executed to the remote server based on the master cloud profiling information such as remote VM execution time and queuing time. Herein, the mobile devices need to contain very small amount of information in order to increase the battery lifetime of the mobile device. The responsibility of taking offloading decision is shifted to the cloud. Fig 1. presents an overview of its architecture and contains the following components.

3.1.1 Mobile Device. Herein, the core application runs on to the mobile device. There can be multiple mobile devices in our assumed scenario and each device have some offloadable tasks. In the mobile device, it has local execution time (LEC) and user defined maximum allowable time (MAT) for each methods. For each offloadable methods, it has also communication latency. In our scenario, we assume that local execution time, maximum allowable time and communication latency are sent to the local cloud master for selecting optimal remote VM. The informations are received by the local master cloud and the master cloud has the responsibility to select optimal VM.

3.1.2 Master Cloud. The master cloud take the remote VM's execution time and average queuing time for processing a task. Herein, we assume that mobile device send each task's local execution time (LEC), maximum allowable time (MAT) and communication latency information to the master cloud. Then, the master cloud select optimal remote VM for executing the offloadable task. The informations collected form the mobile device and remote VM are periodically collected and stored. The master cloud is composed of two main components namely profiler and scheduler. The functions of these are narrated below. Profiler The profiler stores the profiling information of tasks which are used by the scheduler in order to optimally identify remote VM. The profiler stores VM's execution time and queuing time information. The profiler also stores mobile devices LEC, MAT and communication latencies provided by the mobile device. All these information are used by the scheduler to select the optimal remote VM in order to increase performance of the mobile device. When the same or similar task is executed, the proposed system used the previous stored information. Any wrong measurements may force the proposed system to make wrong decision of selection. The informations stored in the profiler from mobile device and remote resource are used by the scheduler. As selection of the optimal remote VM is an optimization problem, so scheduler solve the mathematical model for selecting remote resource. The model of the optimization problem uses all the profiler information and identify the optimal remote VM. The scheduler invokes the objective function to determine the optimal scheduling for executing offloadable task.

3.1.3 Virtual Machine. The remote resources are responsible for executing the offloadable tasks decided by the master cloud. The master cloud uses remote VM's execution time and queuing time information for selecting optimal VM. After selecting the optimal remote resource, the mobile device offloadable tasks are delivered to the selectable VM via the master cloud and return the results back to the mobile device. In order to schedule to an optimal remote VM, we need to have some tasks on to the proposed master cloud. The master cloud then decide which VM is optimal for executing the tasks considering the VM execution time and average queuing time. In our scenario, we assume that we have some tasks for executing to the VM in the master cloud. For partitioning and delivering the tasks to the master cloud, we use the MpOS: A Multiplatform Offloading System [17] framework. For supporting the code offloading mechanism, the framework partitioned the application into methods level. Therefore, some methods are always executed locally and developers need to annotate [**?**, 18] these methods for executing locally. Locally executable methods have the following properties [19, 20, 21]:

(1) Methods that use mobile device specific hardware such as sensors, camera.
(2) Methods that create user interface of the application.
(3) Methods that executed sensible data where re-execution hamper the actual results.

The local and remote executable methods are annotated as @LocalTask and @RemoteTask wherein methods that are locally annotated are always executed locally but remote annotated methods can be executed either locally or remotely depending on variable parameters.

3.2 Optimal Task Allocation

The task allocation to remote resources on the cloud should be an optimal one so that the quality of experience of the users can be enhanced. The major factors impacting this decision are the communication latency from the mobile to the cloud, network inconsistency, and queuing time to the VM and resource heterogeneity. In this section, we describe our proposed optimal task allocation model. As selection of optimal remote resource is an optimization problem, no polynomial time solution can be found for this problem.

3.2.1 Problem Statement. The execution of computational expensive to remote virtual machines has become a rising problem

[18, 19]. In this situation for mitigating the problem, it is required to use distributed computing resources for the compute intensive tasks. Mobile cloud computing can be a solution, but in that case the entire application execution is needed to be done in the cloud. However, it is not feasible as the computational time and cost are increased. Therefore, it is required to partition the application and identify the optimal remote resource for executing the task. In our case, we have assumed that application codes are already partitioned and sent to the master cloud. The task allocation problem for mobile cloud computing consists of a set of remote resources which may be a public server, master cloud or VM's and a set of tasks that has to be offloaded to the remote resource. The remote resource has different configuration and different queuing time. We assume that the communication latency for sending and receiving offloadable data, local execution and maximal allowable time for a tasks are known to the master cloud beforehand. We have to select a optimal VM from different type of VM in order to offload tasks considering the VM's execution and queuing time. For this reason, we need to minimize the execution time and queuing time of a VM for a task in order to increase performance. One task can be allocated to only one server from the set of remote resources is prerequisite for each tasks.

3.2.2 Mathematical Model for Task Allocation. The problem of selecting the optimal remote computing virtual machine to execute a task from mobile device is a challenging task because multiple variable parameters associated with this. The Quality of Service (QoS) of application enforces a prerequisite to execute the task within the maximum allowable time. Mobile devices have local execution time and communication latency C known by the master cloud. As there are different types of VM and different queuing time, one task takes different time to execute and having different queuing time on different VM. To enhance the quality of experience of the users, the response time of the tasks should be kept as minimum as possible. The execution time and queuing time of the VM has a great impact to execute a task. For this reason, the execution time and queuing time of the VMs should be minimum.

Calculation of Queuing Time

In our scenario, we assume that the master cloud has some offloadable task and need to select optimal VM for executing the task. The proposed system considers the heterogeneity of the remote VM. We also consider that each VM has a task queue for incoming task. The VM receives tasks and departure only one at any given time stamp and it can be modeled as M/M/1 queuing model. For calculating the average queuing time, we use the exponential weighted moving average (EWMA). The average queuing time is calculated using the following equation [29].

$$Q^i_{avr} = (1-\alpha) * Q^{i-1}_{avr} + \alpha * Q^i_{curr} \quad (1)$$

Calculation of Communication Latency and Execution Time

In offloading, compute intensive tasks transfer a large amount of data to the master cloud and the result has received back to the mobile device from the master cloud. Herein we assume that transferring and receiving time of offloadable tasks from the mobile cloud to selectable VM is very small because remote resources have high computation power. The communication latency is the sum of the transferring and receiving time of offloadable tasks. The transferring and receiving time vary based on the signal strength between mobile device and access point and different signal strength uses different date rate for sending and receiving data.

Herein, the value of transferring and receiving time is greatly affected by the received signal strength indicator called RSSI value. When the RSSI value from the access point to which mobile devices are connected is high then the achievable data rate for the connection would be high. As a result, the time to send and receive application data will be low and vise versa. Mobile device's ith task communication latency is represented as

$$C_i = t^a_i + t^r_i \quad (2)$$

Let, a be the amount of data bits to be sent and r be the amount of received data for an application having instruction count, I. D be the achievable data rate of the mobile device for an access point attached to the remote server.

Suppose we have a set of T tasks in an application. Also suppose that we have a set of V VM's. In mobile device, there are local execution time and maximum allowable time for each methods which are represented as ϕ and τ respectively. Let we have communication latency, C for each task and a execution time of VM, E for each task assignment. The remote executable VM has a average queuing time, Q. The following objective function select a optimal VM from a set of VM where both execution time and queuing time are minimized where i denotes the tasks to be offloaded to the cloud and j denotes the remote executable VM.

Herein,
E_{ij} =Execution time of i^{th} task to j^{th} VM
Q_j = Average queuing time to j^{th} VM
C_i =Communication latency for i^{th} tasks which is specially,
$C_i = t^a_i + t^r_i$
t^a_i = Time to transfer a task of size a
t^r_{ij} = Time to receive a task of size r
τ_i = Maximum allowable time of i^{th} task
ϕ_i = Local execution time of i^{th} task on mobile device
Let,

$$x_{ij} = \begin{cases} 1, & M_i \text{ task is assigned to } S_j \text{ server.} \\ 0, & \text{otherwise} \end{cases}$$

The research problem can be presented as the following formula.
min $\sum_{ij}(E_{ij}x_{ij} + Q_j x_{ij})$
s.t.
$C_i + \sum_j (E_{ij} + Q_j x_{ij}) \leq \tau_i \qquad \forall_i$
$\sum_j x_{ij} = 1 \qquad \forall_i$

where
$x_{ij} \in \{0, 1\} \qquad \forall i, j$
$C_i, \tau_i, \phi_i \in \mathbb{R}^+ \qquad \forall i$
$E_{ij}, Q_j \in \mathbb{R}^+ \qquad \forall i, j$

Here, the objective is to minimize the execution time and queuing time of i^{th} task to j^{th} VM which is formulated as a single objective integer linear programing problem.

Herein, we assume that mobile devices communication latency, local execution time and maximum allowable time are sent to the master cloud profiler. The profiler in the master cloud uses these information to select remote executable VM for a task. The scheduler of the master cloud solves the optimization problem to identify the VM for a task. It is not possible to use polynomial time algorithm to find an exact solution to the problem for a real time large size applications. Therefore we will try to estimate greedy algorithm having complexity $O(n)$ or $O(logn)$.

3.3 Proposed Greedy Algorithms

The research describes three different greedy algorithm task allocation based on random selection, task allocation based on lowest capacity, task allocation based on highest capacity. Herein, Algorithm 1 present a variation of Monte Carlo Methods that have been applied and algorithms 2 and 3 present greedy based algorithms that have been applied. The algorithms are used to find total time for scheduling a task to optimal remote VM.

3.4 Random Selection of Servers

The objective of Algorithm 1 is to assign a task to randomly selectable VM from a set of VM such that each task is assigned to exactly one VM. The algorithm takes a set of VMs, set of tasks, communication latency, execution time and queuing time of tasks as input and provide the total time as output. Herein, steps (3-6) generate a random number from 1 to the total number of VM type, k and assign that task, t to randomly selectable VM. In step 7, total time variable, λ initialized as zero. Next steps (8-14) derive the value of λ for all tasks.

Algorithm 1: Task Allocation Based on Random Selection

Input : Set of VM V, Set of tasks T, communication latency C, execution time E, queuing time Q, number of VM type k, maximum allowable time τ
Output: totalTime λ

1 Begin;
2 We have set of servers, $V = \{V_1, V_2, ..., V_k\}$;
3 **for *each*** $m \in T$ **do**
4 | generates a random number , $i \in \{1, k\}$;
5 | $V_i \leftarrow V_i \cup \{m\}$;
6 **end**
7 $\lambda \leftarrow \emptyset$;
8 **for *each*** $V_i \in V$ **do**
9 | **for *each*** $m \in T$ **do**
10 | | **if** $(C[m] + E[V_i, m] + Q[V_i]) \leq \tau$ **then**
11 | | | $\lambda += E[V_i, m] + Q[V_i]$;
12 | | **end**
13 | **end**
14 **end**
Output: totalTime, λ

3.5 Task Allocation Based on Lowest Execution Time

In Algorithm 2, tasks are first assigned to the VM having minimum execution time with the constraint that each task is assigned to exactly one server. Set of VMs, tasks, communication latency, execution and queuing time of tasks served as input to the algorithm and provide the total time as output. In step 2, δ describes different type of k VM. In steps (3-5), each task is assigned to the remote VM by master cloud having minimum δ value until all tasks assignment are completed. Total time variable, λ initializes to zero in step 6. Next steps (7-11) calculates the value of λ for all tasks.

3.6 Task Allocation Based on Average Queuing Time

Algorithm 3 task is to assigned tasks to VM having lowest queuing time with the constraint that each tasks are assigned to exactly one VM. The algorithm takes set of VM, set of task, communication latency, maximum allowable time, queuing and execution time of tasks from environment as input and provide the total time as output. In step 2, δ describes the different type of k VM. In steps (3-5), each task is assigned to the remote server having minimum value of δ until all task assignment is completed. Step 6 initialized the total time variable, λ to zero. Next steps (7-13) calculates the value of λ for all tasks where total time must be within the maximum allowable time.

Algorithm 2: Task Allocation Based on Lowest Execution Time

Input : Set of VM V, Set of tasks T, communication latency C, execution time E, queuing time Q, number of VM type k
Output: totalTime λ

1 Begin;
2 $\delta \leftarrow \delta_1, \delta_2, \delta_3....\delta_k$ is the different type of VM, $V_i, i \in \{1, k\}$;
3 **for *each*** $m \in T$ **do**
4 | Assign $t \in T$ to VM V_i where δ_i is lowest execution time; ;
5 **end**
6 $\lambda \leftarrow \emptyset$;
7 **for *each*** $V_i \in V$ **do**
8 | **for *each*** $m \in T$ **do**
9 | | $\lambda += E[V_i, m] + Q[V_i]$;
10 | **end**
11 **end**
Output: totalTime λ

Algorithm 3: Task Allocation Based on Average Queuing Time

Input : Set of VM V, Set of tasks T, communication latency C, execution time E, queuing time Q, number of VM type k, maximum allowable time τ
Output: totalTime λ

1 Begin;
2 $\delta \leftarrow \delta_1, \delta_2, \delta_3....\delta_k$ is the different type of VM, $V_i, i \in \{1, k\}$;
3 **for *each*** $m \in T$ **do**
4 | Assign $m \in T$ to VM V_i where δ_i is average queuing time; ;
5 **end**
6 $\lambda \leftarrow \emptyset$;
7 **for *each*** $V_i \in V$ **do**
8 | **for *each*** $m \in T$ **do**
9 | | **if** $(C[m] + E[V_i, m] + Q[V_i]) \leq \tau$ **then**
10 | | | $\lambda += E[V_i, m] + Q[V_i]$;
11 | | **end**
12 | **end**
13 **end**
Output: totalTime, T

4. PERFORMANCE EVALUATION

Offloading makes the mobile devices possible to execute tasks remotely and improve their performance. It highlights the detail performance evaluation of our proposed greedy algorithms. Initially, the experimental environment is illustrated. Next, the results of the experiments are highlighted to evaluate the proposed algorithms.

4.1 Environment Setup

The experimental environment set up for this experiment is composed of one mobile devices supporting android platform and four laptops of different models VM. The detailed device specification for the experimental environment is listed in Table 1. During execu-

Table 1. : Device Specifications

Equipment	Specification
Mobile Device	Android KitKat Sony Xperia C3 Dual, Qualcomm MSM8926 Snapdragon 400, Quad-core 1.2 GHz Cortex-A7, 1 GB RAM
VM1	Windows 7 64 bit, Quad Core, 1.2 GHz and 1GB RAM
VM2	Windows 7 64 bit, Intel Core i3-4010U, 3 cores, 1.70 GHz and 2GB RAM
VM3	Windows 8.1 Pro 64 bit, Intel Core i5-3450U, 5 cores, 3.10 GHz and 4GB RAM
VM4	Windows 8.1 Pro 64-bits, Intel Core i7-5500, 7 cores, 2.40 GHz and 8GB RAM

tion of the experiments, the tasks are executed on the mobile device first to determine the local execution time. The users of the application are defined the maximum allowable time for each of the tasks. The local execution and maximum allowable time for each task are displayed in Table 2(a). In our scenario, we assume that the local execution and user defined maximum allowable time information are served to the master cloud and it uses these information for selecting optimal remote resource.

4.2 Execution Time and Task Assignment

In order to calculate the value of the proposed mathematical model objective function value, execution time of each task, communication latency for offloadable task and assignment of tasks to remote resources are required.

4.2.1 Task Execution Time. In task offloading, a tasks can be executed either local mobile device or any other remote resources. The decision of selecting remote resources for executing the offloadable task is taken by the master cloud. As we consider the resource heterogeneity, so a task has different execution time and queuing time. Our objective is to minimize the execution and queuing time of a task. The execution time for each task on different VM is displayed in Table 3 and average queuing time is displayed in Table 4. For calculating the average queuing time we use exponential weighted moving average. Herein, one task complete at a given time stamp from the queue. In this assumption we model the queue as M/M/1 queuing model.

Table 2. : Local Execution and Com Latency

(a) Local Execution and Maxi-mum Allowable Time of Tasks

Task	Local Execution Time	Maximum Allowable Time
m1	16.654	13.323
m2	18.662	14.93
m3	29.203	23.362
m4	23.974	19.179
m5	31.75	25.4
m6	32.443	25.954

(b) Communication Latency for Task

Task	Transfer	Receive	Com Lat ency
m1	1.01	0.49	1.5
m2	0.92	0.67	1.59
m3	1.06	0.71	1.77
m4	1.12	0.79	1.91
m5	0.65	0.44	1.09
m6	0.74	0.58	1.32

4.2.2 Communication Latency. Mobile devices are attested to a particular access point and getting a particular signal strength. For each signal strength, there are fixed data rate for sending and receiving data. In 802.11a2 there are different data rates. In our scenario, transferring and receiving time considered as communication latency and assume that the master cloud has the communication latency information. Table 2(b) displays the communication latency for each of the task which are served to the local cloud master.

4.2.3 Task Assignment. In the proposed mathematical model, x_{ij} is used to identify which tasks are executed to which remote resource. The value for the variable is either 1 if task is remotely executable or 0 if locally executable. The assumption for the variable is that for each task, the summation of all remote VM assignments is equal to 1 because a task can be assigned and executed only once to remote VM. The assignment can be different based on the different algorithm type. $V1$, $V2$, $V3$ and $V4$ represent the different type of VM having different configuration. Task allocation based on random selection of VM used Monte Carlo (MC) approaches. In MC approaches, Random number for each tasks are generated for 1 million times and best possible assignment is used for that task.
In task allocation based on random selection of VM assignment, tasks are randomly assigned to remote VMs having different execution and queuing time as follows:

(1) $m1 \rightarrow V3$
(2) $m2 \rightarrow V4$
(3) $m3 \rightarrow V3$
(4) $m4, m5 \rightarrow V1$
(5) $m6 \rightarrow V2$

In task allocation based on lowest execution time, all tasks are always executed to the VM having lower execution time and these assignments are:

(1) $\{m1, m3, m4, m6\} \rightarrow V4$
(2) $\{m2, m5\} \rightarrow V2$

In task allocation based on average queuing time of VM, task assignments are:

(1) $\{m1, m2, m3, m4, m5, m6\} \rightarrow V4$

4.3 Evaluation

Let, suppose there are six methods in an application and four VM having different execution and queuing time. The research proposed three different greedy algorithms to validate our proposed mathematical model. The algorithms uses three different assignment of tasks to the remote VM. This section presents all those algorithms objective value for three different assignments. For example, suppose $m1$ is assigned to $V4$ server in random assignment. Let the objective value be, 0*5.231 + 0*3.126 + 1*1.356 + 0*0.763 + 11.118 that means for $m1$ task $V3$ execution and queuing time will be the total objective value. Only the VM execution time for each task is displayed in Table 3. For all such tasks and remote resources execution and queuing time are displayed in Table 5.

4.4 Discussion And Findings

To identify which context of remote resources and what network context mostly affected the offloading operation, we proposed a mathematical model to take the decision on scheduling. As identify the optimal remote resources for scheduling the tasks required

Table 3. : Execution Time for Each Task

Task	VM1	VM2	VM3	VM4
m1	5.231	3.126	1.356	0.763
m2	5.268	3.589	1.795	0.843
m3	11.128	8.689	5.37	1.132
m4	10.625	6.354	2.069	0.71
m5	15.269	8.457	4.787	1.25
m6	19.263	12.758	6.856	1.921

Table 4. : Average Queuing Time

Task	VM1	VM2	VM3	VM4
m1	5.31	3.26	1.56	0.63
m2	5.23	3.13	1.36	0.76
m3	5.27	3.59	1.79	0.84
m4	11.12	8.68	5.36	1.13
m5	10.63	6.36	2.08	0.71
m6	15.26	8.45	4.78	1.25

Table 5. : Total Execution and Queuing Time For The Proposed greedy Algorithms

Task	Task Allocation Based on Lowest Execution Time	Random Assignment of Tasks	Task Allocation Based on Average Queuing Time	Local Exe
m1	1.39	2.92	1.39	16.654
m2	3.16	1.6	1.6	18.662
m3	1.97	7.16	1.97	29.203
m4	1.84	21.75	1.84	23.974
m5	6.87	25.9	1.96	31.75
m6	3.17	21.21	3.17	32.443
Obj Value	18.4	80.54	11.93	152.686

real time decision. As a result, three different greedy algorithms are proposed and their objective value from the proposed model are derived. For all those algorithms, assignment to high capacity VM result is better than any other assignments. The algorithm also shows better result than local execution time and maximum allowable time for all tasks.

Fig 2(a) shows the comparison of all tasks based on the execution time on different VM. Fig 2(b) considers not only the execution time but also consider the queuing time of the VM. The result shows that queuing time largely affected the overall result. The proposed greedy algorithm result comparison is displayed in Fig 2(c). From our evaluation, we found that higher capacity VM has less queuing delay and minimizes the task execution time. The result is compared with the local assignment in Fig 2(d). If remote execution save the execution time from the local execution, then the remote execution is beneficial which is shown from our result.

5. CONCLUSION AND FUTURE WORK

The research addressed the issue of selecting optimal resource on cloud for remotely executing the compute intensive tasks from the mobile device. In order to select the resources in an optimal way, it has formulated a mathematical model and analyzed to select the remote executable tasks. The proposed system architecture consists of mobile device, local master cloud and the remote virtual machine. In the mobile device, we assume that communication latency, local execution and maximum allowable time information which are available to the master cloud before taking the scheduling decision. The master cloud stores the remote VM execution time and queuing time information and solves the proposed model to take scheduling decision. It also presents three greedy algorithms to validate the proposed mathematical model. The proposed system does not take into consideration the mobility of the mobile devices and security issues like authentication while performing the offloading operation is the future work.

6. REFERENCES

[1] Denzil Ferreira, Anind K Dey, and Vassilis Kostakos. Understanding human-smartphone concerns: a study of battery life. In *Pervasive computing*, pages 19–33. Springer, 2011.

[2] M Satyanarayanan. Mobile computing: the next decade, in proceedings of the 1st acm workshop on mobile cloud computing & services: Social networks and beyond (mcs), 2010.

[3] Ejaz Ahmed, Abdullah Gani, Mehdi Sookhak, Siti Hafizah Ab Hamid, and Feng Xia. Application optimization in mobile cloud computing: Motivation, taxonomies, and open challenges. *Journal of Network and Computer Applications*, 52:52–68, 2015.

[4] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[5] Mahadev Satyanarayanan. Fundamental challenges in mobile computing. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, pages 1–7. ACM, 1996.

[6] Peng Shu, Fangming Liu, Hai Jin, Min Chen, Feng Wen, and Yupeng Qu. etime: energy-efficient transmission between cloud and mobile devices. In *INFOCOM, 2013 Proceedings IEEE*, pages 195–199. IEEE, 2013.

[7] Jiwei Li, Kai Bu, Xuan Liu, and Bin 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*, pages 39–44. ACM, 2013.

[8] Zohreh Sanaei, Saeid Abolfazli, Abdullah Gani, and Rajkumar Buyya. Heterogeneity in mobile cloud computing: taxonomy and open challenges. *Communications Surveys & Tutorials, IEEE*, 16(1):369–392, 2014.

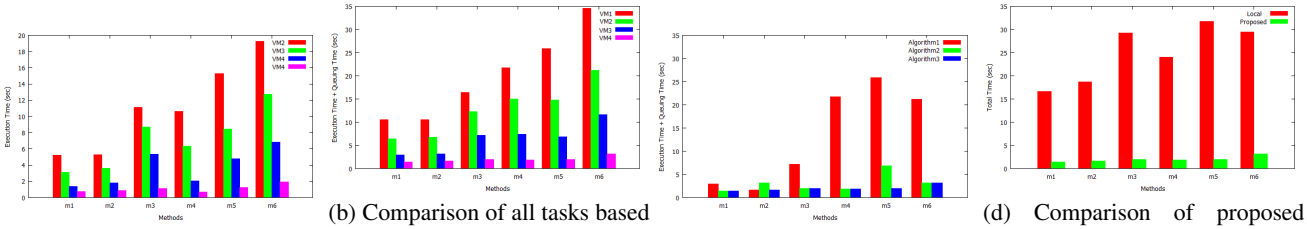
[9] Yaser Jararweh, Loai Tawalbeh, Fadi Ababneh, and Fahd Dosari. Resource efficient mobile computing using cloudlet infrastructure. In *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*, pages 373–377. IEEE, 2013.

[10] Hao Qian and Daniel Andresen. Emerald: Enhance scientific workflow performance with computation offloading to the cloud. In *Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on*, pages 443–448. IEEE, 2015.

[11] Rajesh Balan, Jason Flinn, Mahadev Satyanarayanan, Shafeeq Sinnamohideen, and Hen-I Yang. The case for cyber foraging. In *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, pages 87–92. ACM, 2002.

[12] Rajesh Krishna Balan, Mahadev Satyanarayanan, So Young Park, and Tadashi Okoshi. Tactics-based remote execution

Fig. 2: Discussion and Findings



(a) Comparison of all tasks based on execution time

(b) Comparison of all tasks based on execution time and queuing time

(c) Comparison of algorithms for all tasks

(d) Comparison of proposed mechanism with local mobile device

for mobile computing. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 273–286. ACM, 2003.

[13] Adam J Oliner, Anand P Iyer, Ion Stoica, Eemil Lagerspetz, and Sasu Tarkoma. Carat: Collaborative energy diagnosis for mobile devices. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 10. ACM, 2013.

[14] Gonzalo Huerta-Canepa and Dongman Lee. An adaptable application offloading scheme based on application behavior. In *Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on*, pages 387–392. IEEE, 2008.

[15] Shumao Ou, Kun Yang, and Antonio Liotta. An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems. In *Pervasive Computing and Communications, 2006. PerCom 2006. Fourth Annual IEEE International Conference on*, pages 10–pp. IEEE, 2006.

[16] Balasubramanian Seshasayee, Ripal Nathuji, and Karsten Schwan. Energy-aware mobile service overlays: Cooperative dynamic power management in distributed mobile systems. In *Autonomic Computing, 2007. ICAC'07. Fourth International Conference on*, pages 6–6. IEEE, 2007.

[17] Guangyu Chen, Byung-Tae Kang, Mahmut Kandemir, Narayanan Vijaykrishnan, Mary Jane Irwin, and Rajarathnam Chandramouli. Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices. *Parallel and Distributed Systems, IEEE Transactions on*, 15(9):795–809, 2004.

[18] Deepak Shivarudrappa, M Chen, and Shashank Bharadwaj. Cofa: Automatic and dynamic code offload for android. *University of Colorado, Boulde*, 2011.

[19] Philipp B Costa, Paulo AL Rego, Lincoln S Rocha, Fernando AM Trinta, and José N de Souza. Mpos: a multiplatform offloading system. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 577–584. ACM, 2015.

[20] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62. ACM, 2010.

[21] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *INFOCOM, 2012 Proceedings IEEE*, pages 945–953. IEEE, 2012.

[22] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*, pages 301–314. ACM, 2011.

[23] Roelof Kemp, Nicholas Palmer, Thilo Kielmann, and Henri Bal. Cuckoo: a computation offloading framework for smartphones. In *Mobile Computing, Applications, and Services*, pages 59–79. Springer, 2012.

[24] Mark S Gordon, Davoud Anoushe Jamshidi, Scott A Mahlke, Zhuoqing Morley Mao, and Xu Chen. Comet: Code offload by migrating execution transparently. In *OSDI*, pages 93–106, 2012.

[25] Huber Flores and Satish Srirama. Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning. In *Proceeding of the fourth ACM workshop on Mobile cloud computing and services*, pages 9–16. ACM, 2013.

[26] Moo-Ryong Ra, Anmol Sheth, Lily Mummert, Padmanabhan Pillai, David Wetherall, and Ramesh Govindan. Odessa: enabling interactive perception applications on mobile devices. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 43–56. ACM, 2011.

[27] M Reza Rahimi, Nalini Venkatasubramanian, Sharad Mehrotra, and Athanasios V Vasilakos. Mapcloud: mobile applications on an elastic and scalable 2-tier cloud architecture. In *Proceedings of the 2012 IEEE/ACM fifth international conference on utility and cloud computing*, pages 83–90. IEEE Computer Society, 2012.

[28] Qiufen Xia, Weifa Liang, Zichuan Xu, and Bingbing Zhou. Online algorithms for location-aware task offloading in two-tiered mobile cloud environments. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 109–116. IEEE Computer Society, 2014.

[29] Archan Misra, Teunis Ott, and John Baras. Effect of exponential averaging on the variability of a red queue. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 6, pages 1817–1823. IEEE, 2001.