2016 2nd International Conference on Next Generation Computing Technologies (NGCT-2016)
Dehradun, India 14-16 October 2016

Mobile Cloud Computing Architecture for Computation Offloading

Abhirup Khanna
University of Petroleum and Energy Studies, Dehradun
abhirupkhanna@yahoo.com

Archana Kero
Shri Guru Ram Rai Institute of Technology and Science, Dehradun
archanakero@gmail.com

Devendra Kumar
College of Engineering Roorkee, Roorkee
devmochan@yahoo.co.in

Abstract— **In recent years the Smartphone has undergone significant technological advancements but still remains a low computational entity. Mobile Cloud Computing addresses this problem and provides a solution in form of Mobile Computation Offloading (MCO). Computation offloading is a concept in which certain parts (tasks) of an application are executed on cloud whereas the rest of them on the mobile device itself. MCO turns out to be a great help with respect to resource constrained mobile derives as it allows resource intensive tasks to be executed remotely. Though such procedures save mobile resources but incur communication cost between the mobile device and cloud. Thus, it becomes extremely essential for offloading models to take steps (offloading decisions) in order to augment the capabilities of mobile devices along with reduced execution and communication costs. In this paper, we present an application offloading model which offloads an application based upon the nature and execution pattern of its tasks. We also propose an algorithm that depicts the work flow of our computation model. The proposed model is simulated using the CloudSim simulator. To this end, we illustrate the working of our proposed system along with the simulated results.**

Keywords—Mobile Cloud Computing; Mobile Computation Offloading; Cloud computing; CloudSim

I. INTRODUCTION

In recent times, huge technological advancements have been seen in the smartphone industry. This exponential growth has very well manifested itself in form of the many fold escalation of the application industry since the inception of present day smart phones. Mobile devices are being equipped with more and more storage and computation power, thus allowing developers to create more and more resource intensive applications. Applications which are being designed nowadays are not only more appealing but also more intensive in terms of consumption of device end resources. In spite of such humongous advancements in the resource capabilities of mobile devices to what they where ten years back, they still fail to cope with the resource intensive applications. Today, mobile devices are more advanced but still they still lack behind in terms of resources with respect to PCs and laptops. Researchers where in search of a solution to this problem and then they came across something known as cloud computing. Cloud computing can be defined as a computing model that renders services on pay as you use basis over the Internet. It is the buzz word of the IT industry and has gained huge popularity among the users. It provides services on the fly in a dynamic fashion allowing users to access resources from anywhere in the world [1]. Initially cloud was being accessed by desktop and laptop users but with increase in the smartphone usage its reach went down to the hands of the mobile users. Cloud computing had all those qualities from being scalable to possessing unlimited amount of resources, which made it perfect to handle the short comings of mobile devices. The result to the amalgamation of Cloud computing and Mobile computing gave rise to a new computing model know as Mobile Cloud Computing (MCC) which serves to be a revolutionary idea in order to overcome the short comings of smart phones.

One of the biggest contributions of Mobile Cloud Computing was to resolve the problem pertaining with resource limitations of mobile devices. This was achieved through the concept of Mobile Computational Offloading (MCO), which involved the enhancement of computational capabilities of resource constrained devices by leveraging from the functionalities of cloud computing. Very often it has been seen that the terms “surrogate computing” and “cyber forging” have also been used to address computational offloading [2]. MCO involves the transfer of an application outside a mobile device and allow its execution on a remote computing environment. This remote computing environment can be a virtual machine (surrogate) hosted on a cloud server where the application is allowed to run [3]. Once the execution of the application is over the subsequent results are reverted back to the mobile device. The concept of offloading is optional and is not feasible in cases when excess amount of time and energy are required to send data from the mobile device to the cloud. Significant amount of research has been done in the field of MCO leading to the formation of different architectures and frameworks that support it [4]. The aim of this paper is to present a novel application model for computation offloading wherein an application is divided into multiple tasks which are then offloaded based upon the type of task along with rest of the device end parameters.

The rest of the paper is categorized as follows: Section II elucidates the proposed work wherein the proposed system and its working are explained. Section III discusses the algorithm that depicts the workflow of the entire system, whereas its simulation and results are discussed in Section IV. Finally, Section V concludes the paper.

II. Proposed work

Offloading mobile applications to cloud has gained interest of many researchers and developers thus allowing them to create architectures and frameworks that facilitate code offloading. The basic idea behind the concept of mobile computation offloading is to partition the application explicitly by an application developer in order to execute it remotely. In mobile computation offloading, components of an application (tasks) can be identified as, those that can run locally, those that are to be executed on cloud and those that can be executed either locally or remotely [5]. This means that the same application component may either be allowed to run locally or need to be offloaded depending upon the device end resources. In this section, we propose a computation offloading model that identifies and categorizes tasks as CPU intensive and I/O intensive. As I/O intensive tasks consume limited amount of resources and require impromptu responses from the end user thus they are not considered for offloading by our system. It is the CPU intensive tasks which are to be offloaded by the system depending upon size of the task, number of instructions, execution pattern, execution time and device end parameters.

A. Machine Model

In this subsection we would be discussing the mathematical model for our proposed system. Following is the nomenclature table that describes all the various entities that have been used in this mathematical model.

TABLE I. NOMELCLATURE TABLE

Symbol	Meaning
G	Task Execution Graph
V	Nodes (It represents the various tasks of an application)
E	Edges (They denote the inter-task dependencies)
I	Independent Tasks
D	Dependent Tasks
L	Location of task execution (Mobile device or Cloud)
T(V)	Type of Task (I/O or CPU intensive)
F	Total cost function
t	Communication cost function
e	Execution cost function
U	Mobile Device on which the application is being running
A	The application that may undergo computation offloading
X	Virtual Machine where the offloaded tasks would be executed. It resides at the Cloud end.
O	Offloading factor (0 or 1)
ß	Battery consumption (%)
T	Time consumption (seconds)
CM	Cost Matrix; a matrix representing battery consumption and total execution time
FR	Final Resultant

$G = (V, E)$
$V = \{I \cup D\}$
$L = L_1 \dots\dots\dots L_n$ // Location for n tasks
$L_n \in (M,C)$

$$t_i(L) = \frac{S1+S2}{n} \tag{1}$$

$$e_i(L) = \frac{i}{cpu \times mips} \tag{2}$$

$$F(T_{i,}L) = t_i\ (L) + e_i(L) \tag{3}$$

$V_i \in I$

$$F(T_{i,}L) = t_i\ (L) + e_i(L) + \sum F(T_{j,}L) \tag{4}$$

$V_i \in D \qquad V_j \in (I \cup D)$

$t_i(M) = 0$// Communication cost of running a task locally is 0
$U \rightarrow A^{\sim}$ // Mobile Device running multiple Applications
$A \in (c, b, n, m)$ // Configuration of each application
$X \rightarrow A^{\sim}$ // VM running multiple Applications
$V \in (i, m, s, o)$ // Configuration of a task
$O_0 = O_n = 0$ // First and last tasks are executed locally

$$F(T_{i,}L, \beta, \tau) = CM(\beta, \tau) \tag{5}$$

$$FR = \sum_{i=1}^{n} F(T_{i,}L, \beta, \tau) \tag{6}$$

$$FR_M = \sum_{i=1}^{n} F(T_{i,}\beta, \tau, M) \tag{7}$$

$$FR = \sum_{i=1}^{n} F(T_{i,}L) + \max\left(F(T_{j,}L)\right) \tag{8}$$

$V_i \in$ Task in Series
$V_j \in$ Tasks in parallel

The above proposed mathematical model showcases various equations which talk about the entire process of computation offloading from categorizing a particular task to its execution at the cloud end. We make use of a Directed Acyclic Graph to represent the entire workflow of task execution. The following figure corresponds to one such DAG wherein task execution takes place.

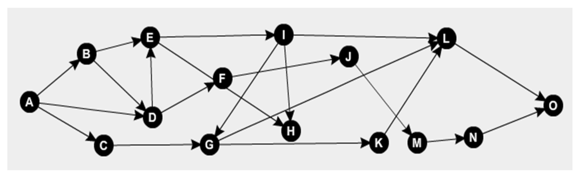


Fig. 1. DAG for Task Execution Graph

As per shown in the figure there are in total fifteen tasks out of which some of them are independent and some share multiple dependencies on others. The first and last tasks ("A", "O") are I/O intensive and are thus executed locally on the mobile device. During computation offloading it is very essential to identify inter tasks dependencies as offloading a parent task could have great impact on the execution of its child tasks. In our proposed system the tasks that are independent are allowed to run in parallel where as tasks which share dependencies are executed sequentially. Following are the descriptions of all the eight equations which have been mentioned in our mathematical model.

- Equation 1 depicts the communication cost function, whose purpose is to calculate the communication cost of a task in terms of battery consumption and time. The time variant of the communication cost function is zero when a task is executed locally on a mobile device.

- Equation 2 depicts the execution cost function, whose purpose is to calculate the execution cost of a task in terms of battery consumption and time.

- Equation 3 depicts the total cost function for all the independent tasks. Independent tasks are those tasks which do not relay on inputs from other tasks.

- Equation 4 depicts the total cost function for all the dependent tasks wherein $\sum F(T_{j,}L)$ represents the total cost of all the parent tasks. Dependent tasks are the ones which require inputs from their parent tasks in order to undergo successful execution.

- Equation 5 depicts the total cost function wherein values such as battery consumption (in percentage) and time (seconds) are rendered from the Cost Matrix.

- Equation 6 depicts the Final Resultant which is a factor used to determine the total cost of executing all the independent tasks of a particular application.

- Equation 7 depicts the Final Resultant of an application execution wherein all tasks of the application where executed locally on the mobile device. In this case, computation offloading hasn't taken place and none of the tasks have been offloaded.

- Equation 8 depicts the Final Resultant of all the tasks (independent & dependent) of an application which has undergone computation offloading.

- Finally, a comparison is made between Final Resultants of Equation 7 and Equation 8. Offloading is only considered beneficial when the Final Resultant from Equation 8 is less than the Final Resultant of Equation 7, which means that the cost of application offloading is less as compared to cost of running the application locally.

B. *Application Model*

In this subsection, we would be talking about our application model of our proposed system. The model comprises of four actors which communicate among themselves and describe the working of our offloading system. The model is well distributed across multiple computing environments and thus has some of its components running on Cloud whereas the others on the mobile device itself. Following is a block diagram which illustrates the application model.

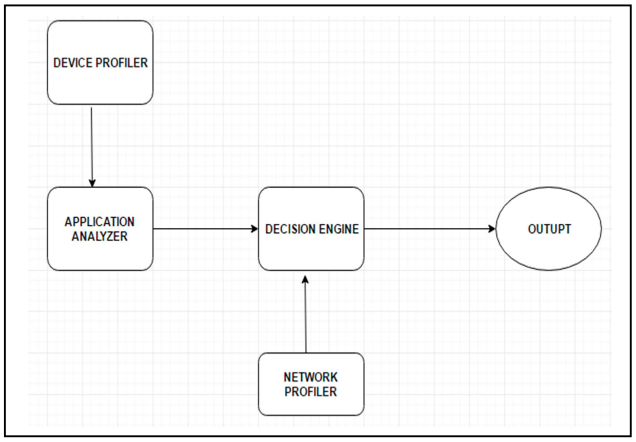


Fig. 2. Offloading Model

- *Device Profiler*: The work of the device profiler is to divide an application into numerous tasks. The division of a task is done on the basis of its being CPU or I/O intensive. A task may either be independent or dependent of its fellow tasks. Special annotations are specified by the device profiler so as the system is able to identify a task in terms of CPU or I/O intensive. The device profiler resides at the mobile end.

- *Application Analyzer*: It calculates the cost of running a task locally or remotely in terms of battery consumption and execution time. It takes into consideration the number of instructions to be executed and memory requirements of a task along with the processing capabilities of the computing environment (mobile device or VM). It resides over the Cloud.

- *Network Profiler*: It calculates the transmission cost of a task in case it needs to be offloaded and resides at the mobile end.

- *Decision Engine*: It receives inputs from both the application analyzer and network profiler. The decision engine resides at the Cloud end. Based upon the inputs it performs certain mathematical calculations and decides whether the application needs to be offloaded or not.

III. ALGORITHM

In this section, we explain the working of our proposed offloading model through the illustration of the algorithm that forms the core for it. The algorithm depicts the functioning of the model by representing the relationship between various actors and the information that they share in form of parameters among themselves.

```
1.  Start
    AZ ≈ Application Analyzer; NP ≈ Network
    Profiler;
2.  Input: An application for computation offloading
    Output: Total Execution Time & Battery
    Consumption
3.  Device Profiler divides the application into
    numerous tasks (N)
4.  for i= 1 to N  do
5.  Type of task $T(V_i)$ = {I/O || CPU};
6.  AZ $\rightarrow$ ei(L);
7.  NP $\rightarrow$ ti(L);
8.  CM($\beta$, $\tau$) = { ei(L) & ti(L)};
9.  Total cost function is computed call (F(T(V)i, L));
10. DE $\rightarrow$ FR(T, $(V_i)$, L);
11. compute (O) = { 0|| 1};
12. if (O==0)  then
     run locally;
13. else
    run remotely;
14. endfor
15. End
```

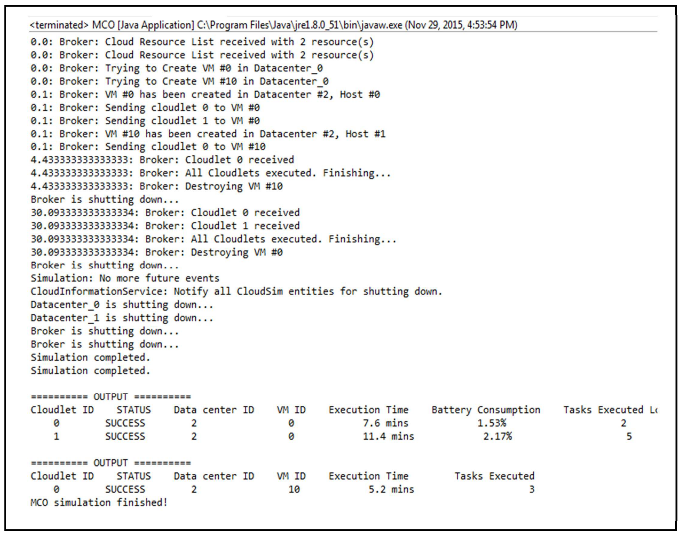
IV. SIMULATION & RESULTS

The above mentioned algorithm is implemented on the CloudSim framework. CloudSim [6] is a simulation toolkit which comprises of various predefined classes that provide a simulation environment for Cloud computing. It is a java based simulation toolkit and can be implemented using Eclipse or NetBeans IDE. In our case we would be using the eclipse IDE. To run CloudSim on eclipse, we first need to download the eclipse IDE and install it. After successful installation of eclipse IDE, download the latest CloudSim package, extract it and import it in eclipse. Talking of our proposed work we have created our own classes in CloudSim and have portrayed our algorithm in form of java code. The following are the screenshots that depict the working of our algorithm on the CloudSim framework.

Following is the parameter table that exhibits all the various parameters that have been considered during simulation.

TABLE II. PARAMETER TABLE

Parameter	**Value**
Application(s)	1
Total Tasks	15
CPU Intensive Tasks	11
I/O Intensive Tasks	4
Size of the application	200MB
Network Bandwidth	1.5MBps



```
<terminated> MCO [Java Application] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (Nov 29, 2015, 4:53:54 PM)
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Cloud Resource List received with 2 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #10 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #0
0.1: Broker: VM #10 has been created in Datacenter #2, Host #1
0.1: Broker: Sending cloudlet 0 to VM #10
4.433333333333333: Broker: Cloudlet 0 received
4.433333333333333: Broker: All Cloudlets executed. Finishing...
4.433333333333333: Broker: Destroying VM #10
Broker is shutting down...
30.093333333333334: Broker: Cloudlet 0 received
30.093333333333334: Broker: Cloudlet 1 received
30.093333333333334: Broker: All Cloudlets executed. Finishing...
30.093333333333334: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Datacenter_1 is shutting down...
Broker is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Execution Time    Battery Consumption    Tasks Executed L
    0         SUCCESS         2              0           7.6 mins           1.53%                     2
    1         SUCCESS         2              0           11.4 mins           2.17%                     5

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Execution Time        Tasks Executed
    0         SUCCESS         2              10           5.2 mins                   3
MCO simulation finished!
```

Fig. 3. CloudSim Simulation

The above mentioned figure depicts the working of our offloading model. Some pre-defined functions such as createCloudlet(), createBroker() and createDatacenter() from CloudSim have been used extensively throughout this experiment for creating various entities such as Cloudlets (Application), Virtual Machines, broker and Datacenter. In this experiment, we consider two computing environments which are represented by VM #0 (mobile device) and VM #10 (Cloud). Two applications (Cloudlet #0 & Cloudlet #1) can also be seen running on VM #0 out of which one application Cloudlet # 0 undergoes computation offloading resulting in certain tasks of it being executed on Cloud (VM #10). During the simulation of our proposed system various metrics reported by CloudSim were collected. The results of the simulation are demonstrated using three different parameters namely, execution time, battery consumption and tasks executed. In first case both the applications are allowed to run locally whereas in second case Cloudlet #0 is offloaded resulting in reduced battery consumption and execution time.

Furthermore we performed an experiment wherein we steadily increased the number of tasks being offloaded and studied the execution pattern of the respective application. Following is the graph which showcases variation in total execution time (execution time + communication time) of an application when its number of offloaded tasks kept on increasing.

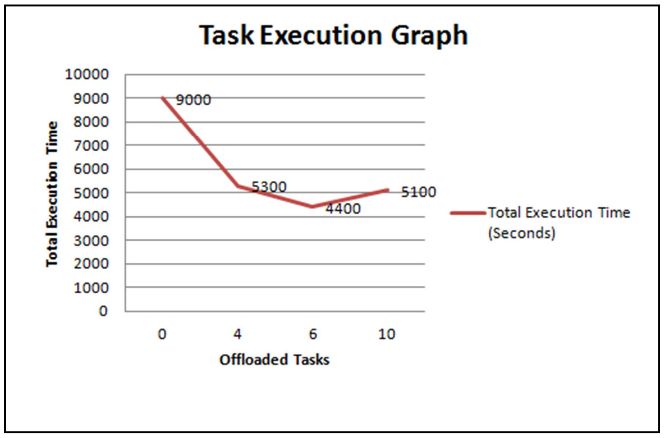


Fig. 4. Experiment Result

The above graph showcases the variation in total execution time of an application in seconds. The X axis represents the number of offloaded tasks whereas the Y axis represents the execution time. The graph clearly depicts how the total execution time of an application started to reduce from 9000 to 5500 and finally 4400 seconds by increasing the number of offloaded tasks. But a sudden rise in execution time can be seen once the number of offloaded tasks were increased from 6 to 10. This experiment makes it very much evident that offloading does reduce execution time and is beneficial in most of the cases but can even turn out to be expensive if applied extensively.

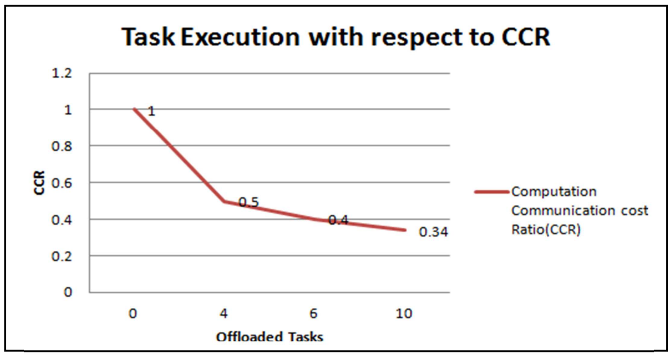


Fig. 5. Experiment Result

We also make use of CCR (Computation to Communication cost Ratio) in order to evaluate our experiment. The above mentioned graph depicts how the CCR continued to decrease even while there was an increase in number of offloaded tasks. These results are very much different than that of the previous graph which showcased an increase in execution time with increase in offloaded tasks after a certain value.

V. CONCLUSION

In this paper, we present an optimal computation offloading model that divides an application into multiple tasks and offloads them on the basis of their nature and execution pattern. The offloading model that we propose is based on the principles of Mobile Cloud Computing. During application offloading the intended model takes into consideration the communication cost along with the execution cost of running a task on cloud. Tasks running on cloud or locally on the mobile device are in complete sync with one another and exchange information that may act as an input for one and output for the other.

References

[1] Khanna, A. (2015, September). RAS: A novel approach for dynamic resource allocation. In Next Generation Computing Technologies (NGCT), 2015 1st International Conference on (pp. 25-29). IEEE.

[2] Enzai, N. I. M., & Tang, M. (2014, April). A taxonomy of computation offloading in mobile cloud computing. In Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on(pp. 19-28). IEEE.

[3] Kovachev, D., Yu, T., & Klamma, R. (2012, July). Adaptive computation offloading from mobile devices into the cloud. In 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (pp. 784-791). IEEE.

[4] Khanna, Abhirup, Sarishma. (2015). *Mobile Cloud Computing: Principles and Paradigms*. IK International.

[5] Dinh, H. T., Lee, C., Niyato, D., & Wang, P. (2013). A survey of mobile cloud computing: architecture, applications, and approaches. Wireless communications and mobile computing, 13(18), 1587-1611.

[6] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1), 23-50.

[7] Jiang, Y., He, J., Li, Q., & Xiao, X. (2014, December). A dynamic execution offloading model for efficient mobile cloud computing. In 2014 IEEE Global Communications Conference (pp. 2302-2307). IEEE.

[8] Cheng, Z., Li, P., Wang, J., & Guo, S. (2015). Just-in-time code offloading for wearable computing. IEEE Transactions on Emerging Topics in Computing, 3(1), 74-83.

[9] Shiraz, M., Sookhak, M., Gani, A., & Shah, S. A. A. (2015). A study on the critical analysis of computational offloading frameworks for mobile cloud computing. Journal of Network and Computer Applications, 47, 47-60.