Adaptive Mobile Computation Offloading for Data Stream Applications

Abhirup Khanna
University of Melbourne
Melbourne, Australia
abhirupkhanna@yahoo.com

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

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

Amit Agarwal
University of Petroleum & Energy Studies
Dehradun, India
aagarwal@ddn.upes.ac.in

Abstract—**In recent times mobile applications are becoming increasingly rich in terms of the functionalities they provide to the end users. Such applications might be very popular among users but there execution may result in draining of many of the device end resources. Mobile Cloud Computing (MCC) provides a better way of executing such applications by offloading certain parts of the application to cloud. At the first place, computation offloading looks quite promising in terms of saving device end resources but eventually turn out to be expensive if carried out in a static manner. Changes in device end resources and environment variables may have huge impacts on the efficiency of offloading techniques and may even reduce the quality of service for applications involving the use of real time information. In order to overcome this problem, we propose an adaptive computation offloading framework for data stream applications wherein applications are partitioned dynamically followed by being offloaded depending upon the device end resources, network conditions and cloud resources. We also propose an algorithm that depicts the work flow of our computation model. The proposed model is simulated using the CloudSim simulator. In the end, we illustrate the working of our proposed system along with the simulated results.**

Keywords—Mobile Cloud Computing; Mobile Computation Offloading; Adaptive Partitioning; Data Stream Applications

I. INTRODUCTION

Mobile cloud computing is an emerging concept, whose aim is to provide better services to mobile users without draining the resources of their devices and burdening them with extra cost of data and energy. 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. Presently most applications fall under offline category, where resources are local and data is downloaded from backend systems. In case of online applications data is mostly at users perusal and web technologies serves as powerful alternatives to inhouse applications. In both these cases to get optimum output system has to be adaptive to respond as per the changes in the mobile ecosystem. To achieve the desired outcome mobile devices can offload to any of the computational infrastructure be it the virtual machines or cloudlets depending upon the particular demand of an application. Here the main concern is to maximize the performance of a smartphone application and conserve energy at the same time. However, mobile cloud computation works in a heterogeneous environment with each client having special demands while using different mobile devices. This requires dynamic partitioning and remote execution, for which some use Alfredo framework[1] to distribute application modules between mobile device and the server. Same way R-OSGi[2] is utilised for interaction between virtual machines in some systems. Otherwise weblets[3] can be used for dividing an application into elastic components for dynamic execution. The weblets are not constrained to one programming language thus allowed to be used for wider range of applications. Process migration is also an important aspect for allowing seamless transition of individual processes without affecting the performance.

In this paper, we have targeted mobile data stream applications and have tried to accommodate such applications on resource-constrained mobile devices by augmenting the execution of such applications and leveraging the cloud resources. We propose an adaptive computation offloading model for data stream applications. The model which we propose is based upon the principles of dynamic code partitioning and works on thread level migration granularity. In our work, we have tried to maximize the performance of data stream applications by reducing the makespan and energy consumption at the same time. Existing models assumes the fact that the mobile user has access to a infinite number cloud resources whereas our proposed model makes offloading

978-15090-6403-8/17/$31.00 © 2017 IEEE

decisions by checking the availability of resources at the cloud-end.

Our remaining paper is organized into following sections: Section II presents our proposed model that talks through the System Architecture and Mathematical Model. Section III talks about the Algorithm for dynamic code partitioning and code offloading. Simulation and Results of the experiments carried out are presented in Section IV. Finally, Section V concludes the paper and suggests future work.

II. PROPOSED WORK

In the past couple of years offloading mobile applications to cloud has gained interest of many researchers and developers thus allowing them to create architectures and strategies that facilitate code offloading[5]. The concept of mobile computation offloading can be applicable in two ways i.e. through static or dynamic code partitioning. Static partitioning involves explicit partitioning of an application by an application developer in order to execute it remotely, whereas in dynamic partitioning the application partition's itself in an automated manner without any human intervention depending upon the execution environment variables. In mobile computation offloading, components of an application can be categorized as, those that are to be executed locally, those that are to be executed on cloud and those that can be executed either locally or remotely [6]. 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. Talking of computation offloading migration granularity plays an important role in determining the efficiency of the offloading model[7]. The lower the granularity more will be the accuracy in determining the execution pattern of the application. In this section, we propose an adaptive computation offloading model for data stream applications. The model which we propose is based upon the principles of dynamic code partitioning and works on thread level migration granularity. The application threads are further categorized as CUP intensive and I/O intensive tasks. Throughout this section we would be using the word task for denoting an application thread. 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, num of instructions, execution pattern, execution time and device end parameters. The offloading objective of our model will be to insure performance enhancement in terms of makespan along with energy conservation.

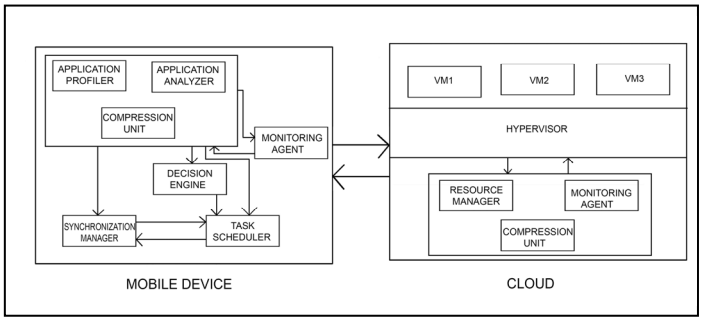

Fig. 1. System Architecture

A. Application Model:

In this subsection, we would be talking about the application model of our proposed system. The model comprises of eight different entities which communicate among themselves and describe the working of our offloading system. The following figure illustrates the application model.

- **Device Profiler**: The work of the device profiler is to divide an application into numerous tasks. The division of tasks is done on the basis of its being CPU or I/O intensive. The device profiler will also identify the execution pattern of a particular set of tasks. A task may either be independent or dependent of its fellow tasks but a particular set of dependent tasks may have an independent execution pattern with respect to another set of dependent tasks. This type of a categorization done by the devise profiler turns out to be extremely useful when offloading multiple tasks at the same time. 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.

- **Application Analyzer**: The work of the application analyzer is to calculate the execution cost of running a task locally or remotely in terms of battery consumption and execution time. The cost of application execution is divided into two parts, computation and communication cost. Communication cost involves cost relating to exchange of task data between mobile device and Cloud. For this purpose the application analyzer communicates with the Compression Unit as it is the one which gives the exact value of data which needs to be transmitted. In order to determine the execution cost, the application analyzer takes into consideration a number of factors such as, number of instructions to be executed, bandwidth of the network connection, size of a particular task along with the processing

978-15090-6403-8/17/$31.00 © 2017 IEEE

capabilities of both the computing environments i.e. Mobile device and Cloud.

- **Compression Unit**: It compresses the task data (code data + computation data) prior to offloading. The compression unit ensures a compression ratio which ranges between 10 to 15. Further details about the algorithms being used for data compression are provided in the section for Mathematical Model.

- **Decision Engine**: It receives inputs from the device profiler and application analyzer. Based upon the inputs it performs certain mathematical calculations and decides whether a task needs to be offloaded or not. It further communicates its decision with the task scheduler.

- **Task Scheduler**: The work of the task scheduler is to schedule the execution pattern of a particular task or a set of tasks. It receives input from the decision engine and accordingly queues the tasks. It also communicates with the application analyzer for knowing the execution pattern of a task. Depending upon the device end parameters the task scheduler may even alter the execution pattern for multiple set of tasks. It may couple a set of tasks which may not be initially relatable based upon their offloading factor and execution cost. The main aim of the task scheduler is to ensure efficient task execution with minimum utilization of resources. Special annotations are specified by the task scheduler so as the system is able to identify a task execution in terms of its execution location.

- **Synchronization Manager**: The role of the synchronization manager is to identify integration points for tasks having different execution locations. It combines output data of multiple tasks executed on different locations and provides it as an input for some other task. It communicates with the task scheduler for knowing the execution location and execution pattern of a task. It also interacts with the application analyzer for knowing the dependencies a task shares with other tasks. It is the synchronization manager which ensures that there is no hindrance in task execution and all the tasks execute in a synchronous manner.

- **Resource Manager**: It is an entity that works at the Cloud end. It is responsible for allocating and managing all the resources that are being rendered to the end user for its application execution[8]. It is the resource manager which communicates with the device Monitoring Agent and dynamically increases or decreases the availability of cloud resources in order to stabilize the execution cost even when there are significant variations (decrease) in the mobile end resources. It also communicates with the Cloud Monitoring Agent in order to know the threshold value for the cloud resources that can be allocated to a particular mobile user.

- **Monitoring Agent**: The work of the monitoring agent is to monitor availability of resources at both the computing environments. Our proposed model comprises of two separate monitoring agents one operating at the mobile device and the other at the Cloud end. Both the monitoring agents are in continuous communication with one another. The monitoring agent of the mobile device transmits information regarding the device end parameters to its counterpart at the Cloud end. This information is further passed on to the resource manager which amends the allocation of resources in order to maintain a stable overall execution cost.

B. Mathematical 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. NOMENCLATURE TABLE

Symbol	Meaning
N	Number of Tasks in an Application
i	Number of instructions in a thread
T	Denotes a task(thread)
B	Bandwidth Required
B	Current Bandwidth
A˜	Denotes an Application
V	Denotes a Virtual Machine
A	Set of Applications
δ	Computational power of a Mobile Device
C	Required Computational power
M	Denotes a Mobile Device
EA	Energy consumption by an application
ET	Energy consumption by a task
D	Uncompressed Data
D˜	Compressed Data
KT	Makespan for a Task
ΨT	Transmission cost function for a Task
θT	Computation cost function for a Task
FT	Total cost function for a Task
RT	Compression cost function for a Task

978-15090-6403-8/17/$31.00 © 2017 IEEE

G= (V, E)
$T_U \in (0,1)$ //type of task i.e. 0 for I/O intensive, 1 for CPU intensive
$T_X \in (0, 1)$ //task execution pattern.0 for Series, 1 for parallel
$\gamma_T \in (0, 1)$ //location of task execution. 0 for Mobile Device, 1 for Cloud
$O \in (0, 1)$ //Offloading Factor. 0 for not offloading, 1 for offloading

$A \rightarrow (A\tilde{})$
$M \rightarrow A$
$V \rightarrow A$
$M \in (\delta, S, B)$
$A\tilde{} \in (N, C_A, E_A)$
$T \in (I, T_U, C_T, b, T_X, E_T)$

$$\Psi_T (K_T, E_T) = (D1 + D2) / B \quad (1)$$

Communication cost function, whose purpose is to calculate the communication cost of a task in terms of battery consumption and time.

$$\theta_T (K_T, E_T) = I / \delta \quad (2)$$

Computation cost function, whose purpose is to calculate the computation cost of a task in terms of battery consumption and time.

$$\Psi_T (K_T, E_T) = 0 \quad T_U=1 \quad (3)$$

The above equation specifies that the communication cost function for all I/O intensive tasks will be 0 as they won't be offloaded.

$$O_0 = O_N = 0$$

The above equation specifies that the first and last tasks are I/O intensive and will always be executed locally thus having an offloading factor 0.

$$R_T (K_T, E_T) = D \rightarrow D\tilde{} \quad (4)$$

Compression cost function, it compresses the task data before task offloading. We would be using two compression techniques for our model. The first one being the JPEG Encoding which is a lossy compression technique and the second one is Huffman Encoding which is a lossless compression technique. JPEG Encoding is used for compressing multimedia data whereas Huffman Encoding is used in case of coded data. The combined compression ration ranges between 10 to 15.

$$F_T (K_T, E_T) = \theta_T (K_T, E_T) \quad (5)$$
$T_X = 1, \gamma_T = 0$

Total cost function of an independent task running locally. As the task execution takes place at the mobile device thus $\Psi_T (K_T, E_T) = 0$

$$F_T (K_T, E_T) = \theta_T (K_T, E_T) + \Psi_T (K_T, E_T) + R_T (K_T, E_T) \quad (6)$$
$T_X = 1, \gamma_T = 1$

Total cost function of an independent task running remotely.

$$F_T (K_T, E_T) = \theta_T (K_T, E_T) + \max(\theta_T (K_T, E_T)) \quad (7)$$
$T_X = 0, \gamma_T = 0 \quad T_X = 1, \gamma_T = 0$

Total Cost Function of a dependent task running locally.

$$F_T1 = \theta_T (K_T, E_T) + \Psi_T (K_T, E_T) + R_T (K_T, E_T) \quad (8)$$
$T_X = 0, \gamma_T = 1$

$$F_T2 = \max(\theta_T (K_T, E_T) + \Psi_T (K_T, E_T) + R_T (K_T, E_T)) \quad (9)$$
$T_X = 1, \gamma_T = 1$

$$F_T (K_T, E_T) = FT1 + FT2 \quad (10)$$

Total Cost Function of a dependent task running remotely.

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**
2. **Input**: An application for computation offloading
   Output: Makespan & Battery Consumption
3. Device Profiler divides the application into numerous tasks (N)
4. **for** i←1 to N **do**
5. $T_X \leftarrow \{0 \parallel 1\}$;
6. $\gamma_T \leftarrow \{0 \parallel 1\}$;
7. **if** $T_X == 1$ AND $\gamma_T == 0$ **then**
8. $\Psi_T \leftarrow 0$
9. $F_T (K_T, E_T) \leftarrow$ *costFunction*$(\theta_T (K_T, E_T))$ // Total cost function is computed
10. $O \leftarrow$ *decisionEngine*$(F_T (K_T, E_T))$
11. **end if**
12. **if** $T_X == 1$ AND $\gamma_T == 1$ **then**
13. $F_T (K_T, E_T) \leftarrow$ *costFunction*$(\theta_T (K_T, E_T), \Psi_T (K_T, E_T), R_T (K_T, E_T)))$
14. $O \leftarrow$ *decisionEngine*$(F_T (K_T, E_T))$
15. **end if**
16. **if** $T_X == 0$ AND $\gamma_T == 0$ **then**
17. $F_T (K_T, E_T) \leftarrow$ *costFunction*$((\theta_T (K_T, E_T), \Psi_T (K_T, E_T))_{TX=0}, (\max (\theta_T (K_T, E_T))_{TX=1})$
18. $O \leftarrow$ *decisionEngine*$(F_T (K_T, E_T))$
19. **end if**
20. **if** $T_X == 0$ AND $\gamma_T == 1$ **then**
21. $F_{T1} \leftarrow$ *costFunction*$((\theta_T (K_T, E_T), \Psi_T (K_T, E_T), R_T (K_T, E_T))_{TX=0}$
22. $F_{T2} \leftarrow$ *costFunction*$(\max((\theta_T (K_T, E_T), \Psi_T (K_T, E_T), R_T (K_T, E_T))_{TX=0})$
23. $F_T \leftarrow F_{T1} + F_{T2}$
24. $O \leftarrow$ *decisionEngine*$(F_T (K_T, E_T))$
25. **end if**
26. **if** (O==0) **then**
    run locally;
27. **else**
    run remotely;
28. **end for**
29. **End**

978-15090-6403-8/17/$31.00 © 2017 IEEE

F

IV. SIMULATION & RESULTS

The above mentioned algorithm is implemented on the CloudSim framework. CloudSim[16] 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 either 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.

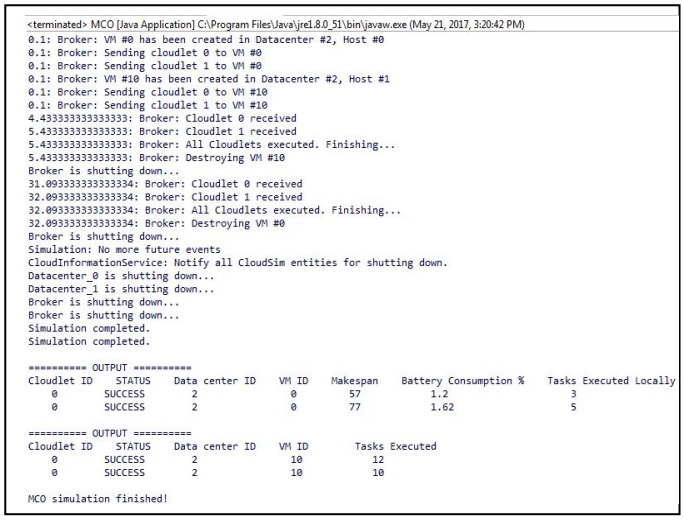

Fig. 2. CloudSim Simulation

The above mentioned screenshot 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, there are two computing environments which are represented by VM #0 (mobile device) and VM #10 (Cloud). Cloudlet 0 is the application that undergoes computation offloading. While performing this simulation we took two instances of the same application with device end parameters changing every time. For the first instance the battery consumption turns out to be 1.2 % and the tasks being executed locally remain 3. Whereas for the second instance the battery consumption increases to 1.62 % and the number of tasks being executed locally add up to 5. This clearly depicts that for the second instance there was a sharp decrease in computation power and network bandwidth with respect to what they were in case of the first instance.

V. CONCLUSION & FUTURE DIRECTIONS

In this paper, we have proposed adaptive offloading model for mobile data stream applications. Our system allows offloading on real-time basis. We have designed an algorithm in which code partitioning is done at thread-level followed by dynamic offloading. Our proposed algorithm tries to enhance the performance of data stream applications in terms of makespan and energy consumption when parameters such as computational load and bandwidth availability are varied. The results of simulated experiments show stability in performance even when resources are scarce. This work can be extended for data-intensive data stream applications. We plan to perform one more task categorization as data-intensive tasks. In future, we intend to incorporate the concept of background augmentation for data stream applications. Ultimately, we envision a system, where, in addition to achieving minimal makespan and energy-consumption, throughput can also be maximized.

References

[1] J. Rellermeyer, O. Riva, and G. Alonso, "AlfredO: An Architecture for Flexible Interaction with Electronic Devices," in Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware (Middleware 2008), ser. Lecture Notes in Computer Science, vol. 5346. Leuven, Belgium: Springer, 2008, pp. 22–41.

[2] J. S. Rellermeyer, G. Alonso, and T. Roscoe, "R-OSGi: Distributed Applications Through Software Modularization," in Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference Conference (Middleware 2007). Newport Beach, CA, USA: Springer, Nov. 2007,pp. 50–54.

[3] Zhang, X., Kunjithapatham, A., Jeong, S., & Gibbs, S. (2011). Towards an Elastic Application Model for Augmenting the Computing Capabilities of Mobile Devices with Cloud Comp-ACM.2011.pdf.

[4] Kumar, K., & Lu, Y. H. (2010). Cloud computing for mobile users: Can offloading computation save energy?. *Computer*, *43*(4), 51-56.

[5] Alam, S., Dewangan, K., Sinharay, A., & Ghose, A. (2016, June). Mobile sensing framework for task partitioning between cloud and edge device for improved performance. In *Computers and Communication (ISCC), 2016 IEEE Symposium on* (pp. 379-384). IEEE.

[6] Khanna, A., Kero, A., & Kumar, D. (2016, October). Mobile cloud computing architecture for computation offloading. In *Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on* (pp. 639-643). IEEE.

[7] Tao, Y., Zhang, Y., & Ji, Y. (2015, March). Efficient Computation Offloading Strategies for Mobile Cloud Computing. In *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on* (pp. 626-633). IEEE.

[8] Liu, J., Ahmed, E., Shiraz, M., Gani, A., Buyya, R., & Qureshi, A. (2015). Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions. *Journal of Network and Computer Applications*, *48*, 99-117.

[9] 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.

[10] Rudenko, A., Reiher, P., Popek, G. J., & Kuenning, G. H. (1998). Saving portable computer battery power through remote process execution. *ACM SIGMOBILE Mobile Computing and Communications Review*, *2*(1), 19-26.

978-15090-6403-8/17/$31.00 © 2017 IEEE

[11] Smit, M., Shtern, M., Simmons, B., & Litoiu, M. (2012, November). Partitioning applications for hybrid and federated clouds. In *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research* (pp. 27-41). IBM Corp..

[12] Verbelen, T., Stevens, T., De Turck, F., & Dhoedt, B. (2013). Graph partitioning algorithms for optimizing software deployment in mobile cloud computing. *Future Generation Computer Systems*, *29*(2), 451-459.

[13] Gupta, N., & Agarwal, A. (2015, March). Context aware mobile cloud computing: review. In Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on (pp. 1061-1065). IEEE.

[14] Meilander, D., Glinka, F., Gorlatch, S., Lin, L., Zhang, W., & Liao, X. (2014, April). Bringing mobile online games to clouds. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on* (pp. 340-345). IEEE.

[15] Dewangan, B. K., Agarwal, A., & Pasricha, A. (2016, October). Credential and security issues of cloud service models. In Next Generation Computing Technologies (NGCT), 2016 2nd International Conference on (pp. 888-892). IEEE.

[16] 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.

978-15090-6403-8/17/$31.00 © 2017 IEEE