International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-4 Number-2 Issue-15 June-2014

A Computation Offloading Framework to Optimize Makespan in Mobile Cloud Computing Environment

Gaurav[1], Nitesh Kaushik[2], Jitender Bhardwaj[3]

Abstract

In the era of continuously evolving applications of mobile devices in our daily routine, the processing capacity is posing as a bottleneck in offering a snappy experience to the users. Despite such impeccable technological advancements coming ever so swiftly in the industry, the resource constraints like processing power still dwarf the performance of certain types of computation-intensive or data-intensive applications. Cloud computing, with features like rapid scalability, ubiquitous network access, on-demand self-service seems to be the right solution to such problems. Mobile cloud computing has created a reverberation in the technology landscape around the world. In this paper, we focus on augmenting execution of mobile applications using cloud resources, more often known as offloading. The approach uses application partitioning, resource monitoring and computation offloading to address the performance or speedup issues. It monitors the available resources both at mobile and cloud side and then adaptively offloads different components of partitioned application to optimize the execution performance of the application. Genetic algorithm is used to find the optimum offloading scenario and the results are evaluated by simulating our approach and comparing it with the all mobile-side execution and all cloud-side execution.

Keywords

Mobile cloud computing, application partitioning, offloading, makespan.

1. Introduction

As the use of mobile devices (such as smart phones, PDAs, etc.) in our routine tasks is increasing day-by-day, they have literally become an important part of our life. For instance, using a smart phone, a user not only receives and makes calls, but also performs various information processing tasks. They are no longer a luxury, but have become essential because of the ubiquitous computing environment growing continuously all around us. Because of the ever evolving technology and depreciating prices, mobile device manufacturers are offering higher than ever specifications in terms of powerful processors, larger memories, multi-network interfaces, a variety of operating systems such as iOS, Android, Windows Mobile etc. But despite all those high end specifications, mobile devices are still unable to go neck-to-neck with the traditional computational devices. They still seem to be underpowered for certain resource-demanding applications [1]. However, cloud computing provides an illusion of infinite computing resources [2]. According to NIST, Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [3]. This cloud model is composed of five essential characteristics, three service models, and four deployment models. Mobile cloud computing is a new platform combining the mobile devices and cloud computing to create a new infrastructure, whereby cloud performs the heavy lifting of computing-intensive tasks and storing massive amounts of data[4]. Mobile applications leverage this IT infrastructure to generate the advantages such as improvement of processing power and storage capacity, extended battery life, improved reliability and scalability etc.

Three approaches[5] have been proposed for mobile cloud applications: 1) extending the access to cloud services to mobile devices; 2) enabling mobile devices to work collaboratively as cloud resource providers[6][7]; 3) augmenting the execution of mobile applications using cloud resources, e.g. by offloading selected computing tasks required by applications on mobile devices to the cloud. This will

Manuscript received June 4, 2014.
Gaurav, Department of CSE, DCRUST Murthal, Haryana, India.
Nitesh Kaushik, Department of CSE, DCRUST Murthal, Haryana, India.
Jitender Bhardwaj, Department of CSE, DCRUST Murthal, Haryana, India.

allow us to create applications that far exceed traditional mobile device's processing capabilities. In this paper, we focus on the third approach. More specifically, we focus on the problem of serving computation or data intensive applications and propose a partitioning offloading approach to speed up their execution and enhance user experience. We propose an approach which offloads the computation of different components of the application to cloud depending upon the current environmental scenario. It monitors a set of mobile, cloud and network parameters such as available resources at mobile device, resources at cloud, available speed and bandwidth of the network in order to decide whether to offload any computation to the cloud or execute it locally on mobile device. Offloading might benefit in terms of execution speed as cloud is a resource rich platform, but the available network bandwidth might be a spoilsport while transferring the data from mobile to cloud for offloading.

In the rest of the paper, section II consists of the review of related research work in mobile cloud computing. Then, we describe the system model in section III followed by section VI discussing mathematical model. In section V, we evaluate the proposed approach and compare the results with the all-cloud execution and all-mobile execution. Finally, section VI consists of the conclusions drawn and some future scope.

2. Related Work

Although, other related work have also included the speeding up of execution by using collaborative approaches, migrating the execution from mobile devices to resource rich platforms. Our work is similar to that of L. Yang [5] and D. Kovachev [8] but is different in the sense that we have considered the SLA negotiated waiting time which is not considered by the previous works.

Huerta-Canepa et al [7] provide the basic framework for creating a virtual cloud with the help of devices falling in close proximity of each other. The framework detects nearby nodes that will remain in the same area or follow the same movement pattern. In scenarios like downloading a description file at a museum, collocation increases the chances of people willing to perform common tasks. To save the resources like energy and processing power, the collocated mobile devices can collaboratively act as a local cloud and split the task into smaller subtasks to be performed on different devices. The results can then be aggregated and shared. The framework might not provide the same amount of benefits as with traditional cloud but preserves several benefits of offloading and omits the need of internet or a connection to traditional cloud platforms. Fernando et al in [9], on the other hand, propose an opportunistic job sharing approach on an ad hoc cloud. It advocates all kinds of local resources (smart phones, PDA, even laptops and PCs) to be used to collaborate in forming the local cloud to achieve a common goal. Their approach is to overcome the resource sparseness, energy consumption and low connectivity problems faced in traditional mobile cloud computing. Sharing of workload is dynamic, proactive and depends on cost model to benefit all participants. SpACCE concept by Tatsuya et al [10] which can be built according to the needs that occur at any given time on a set of personal, i.e., non-dedicated, PCs and dynamically migrate a server for application sharing to another PC. By migrating the server, redundant calculation capacity of PCs can be utilized for creating a SpACCE, where the response time of the application shared among users is improved.

The abovementioned approaches work well for the common tasks but sometimes one particular user might want to execute some application which cannot be distributed among different mobile devices. Works by R. F. Lopes [11], E.Truyen [12], T.Sakamoto [13] advocate for the migration of the executable block of an application to a resource rich platform. Ricky et al [14] proposed offloading the work to a cloud by using the Stack-on-Demand Asynchronous Exception (SOD_AE) mechanism. Here, the work is offloaded by transferring only the current state of the application from mobile to cloud in order to resume further execution at cloud right from the state left from mobile. Another approach proposed by B. Chun et al[15] emphasizes on embedding the complete software stack of mobile device in a virtual machine hired from the cloud provider. The cloudlet architecture proposed by M. Satyanarayanan [1] introduces the concept of two-tier approach for using cloud. The approach helps in lowering the network latencies by introducing a resource rich cloudlet (computer) between mobile device and cloud. The mobile user can use cloudlet with LAN and offload tasks on it instead of using WAN to connect to the distant cloud. If no such cloudlet is available nearby then the traditional cloud is used. Hyrax [6] is also a similar concept which uses mobile devices as nodes to create a mobile cloud. These nodes are considered

as slaves and the jobs from users are divided into independent 'tasks' and these tasks are distributed among these slave nodes.

In the recent past, researchers have been stressing upon offloading techniques, in which the application is executed in parts. Certain parts of it execute on mobile device and the rest on the cloud. Such approaches give better results in terms of both speedup and energy consumption [16]. B. Chun et al [16] proposed CloneCloud concept, which offloads some components of the application on the cloud platform. Cloud platform being resource rich boosts up the performance.

Here, clone components of mobile application reside over the cloud and are executed the same way they would have executed on mobile. Though, the partitioning decisions are made offline, the concept manages to speed up the execution. L Yang in [5] puts forward an offloading approach for data stream mobile applications where accuracy of application is determined by its throughput. The proposed approach makes dynamic offloading decisions based on resource availability at mobile device. The approach determines a critical component of application which takes the maximum execution time among all components. This critical component becomes the deciding component. D. Kovachev in [8] proposes Mobile Augmented Cloud Services (MACS) based on adaptive computation and elasticity in executing blocks of an application. It is aimed at the applications which have some native functions (which are device dependent and must be performed on mobile device itself) and other functionalities of application which are resource-demanding and can be offloaded to get benefits of cloud.

3. System Model

The system architecture of the proposed approach is illustrated in fig. 1. The system primarily consists of eight components, namely a resource manager, a local execution manager, an offload manager, sequential execution tracker, resource monitor, application behaviour generator, optimization solver, cloud offload manager. The first four components reside on the mobile device whereas the rest of the components reside on a middleware. We describe the components in following sub-sections.

A. Resource Manager

The Resource Manager works for monitoring and management of different resources such as processing capacity of mobile, bandwidth of the available network etc. and sends this information to the Resource Monitor residing on the middleware.

B. Local Execution Manager

This component manages the local execution of application components, i.e., the parts of the application running locally on mobile resources.

C. Offload Manager

This module manages the transmission and reception of data for executing the application components at cloud resources. It transmits the data to the Sequential Execution Tracker on middleware in order to make the data available for execution at cloud, and also receives the output data back at mobile side after the execution at cloud side is completed.

D. Sequential Execution Tracker

It works to basically keep the application components in synchronization by communicating with the Cloud Offload Manager as well as Offload Manager on mobile side to offload certain application components on virtual machines running on cloud.

E. Resource Monitor

This component receives the parameters sent by the Resource Monitor at mobile side to help the Application Behaviour Generator to generate suitable partitioning scheme.

F. Application Behaviour Generator

It generates appropriate partitioning scheme for the application so that the local execution cost and remote execution cost can be estimated for each component.

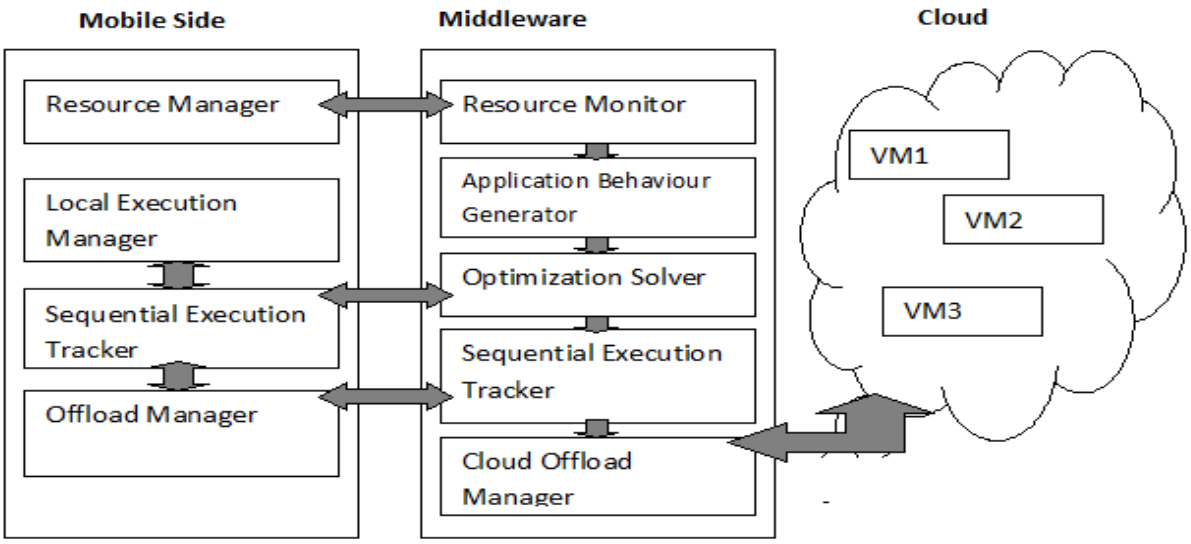


Figure 1: The proposed system model

G. Optimization Solver

The Optimization Solver takes its input from Application Behaviour Generator and generates an optimal offloading scheme in order to minimize the total makespan of the application. This offloading scheme is passed on to both the Sequential Execution Tracker on middleware as well as Sequential Execution Tracker on mobile side.

H. Cloud Offload Manager

It takes the transmitted data from Sequential Execution Tracker in order to offload the execution to a suitable virtual machine running on the cloud. It is basically responsible for providing the input data to the virtual machines as well as for fetching their outputs back.

4. Mathematical Model for Offloading Decisions

The offloading decisions are based upon some mathematical equations. These mathematical equations comprise of the current environmental factors of the scenario, such as the available resources on the mobile device, the average bandwidth of the network currently available for the mobile device and the available resources on the cloud platform for offloaded execution.

Total cost of execution of an application having 'n' number of components can be formulated as:

$$Cost_{app} = \sum_{i=0}^{n} Cost_{comp} \quad \ldots(1)$$

where, i = (1,2,3,…,n) and represent all the n components of the application. We assume that the system is secure and reliable. So, extra overheads required for trust management are negligible.

The cost of execution of a component of the application can be calculated as:

$$Cost_{comp} = \min\{Comp\_Cost_{mob} : Comp\_Cost_{cloud}\} \quad \ldots(2)$$

where, $Comp\_Cost_{mob}$ is the execution cost of component on mobile and $Comp\_Cost_{cloud}$ is the total execution cost of the component on cloud.

The total execution cost of component on cloud is a sum of cost of offloading the corresponding data, cost of executing the component on cloud resources and time spent in waiting queue of virtual machine.

$$Comp\_Cost_{cloud} = Comp\_Cost_{offload} + Comp\_Cost_{exec} + t_{queue} \quad \ldots(3)$$

Given the environmental parameters such as the available mobile resources δ_{mob}, the bandwidth of available network β, the available cloud resources as δ_{cloud}; and the parameters of application such as ψ_i is the number of CPU instructions in component i to process one unit of data, Θ_i is the amount of data required to be offloaded for component i. The cost of executing component i on mobile becomes:

$$Comp_i\_Cost_{mob} = \psi_i/\delta_{mob} \quad \ldots(4)$$

the total cost of executing component i on cloud from equation (3) becomes:

$$Comp_i\_Cost_{cloud} = \Theta_i/\beta + \psi_i/\delta_{cloud} + t_{queue} \quad \ldots(5)$$

And the time spent by the request in cloud virtual machine's queue is:

$$t_{queue} = L_q/\lambda_e \quad \ldots(6)$$

where, L_q is the length of the waiting queue at cloud virtual machine, λ_e is the effective arrival rate of requests at cloud virtual machine, which is further given by:

$\lambda_e = \lambda/[1-P_k]$ …(7)

given that λ is the arrival rate of requests at cloud virtual machine and P_k is the probability that there are *k* requests in the queue.

Integrating all the equations, the total cost of execution of application using this approach becomes:

$Cost_{app} = \sum_{i=0}^{n} (\min \{ (\psi_i/\delta_{mob}) , (\Theta_i/\beta + \psi_i/\delta_{cloud} + t_{queue}) \})$ …(8)

5. Evaluation and Analysis

In this section, we simulate the proposed functionality and analyze the cost of execution of application in our experiments under a variety of different factors related to mobile device, available network as well as cloud platform.

a.) Experimental Setup

Genetic algorithms are used to get the fittest generation of our string for offloading decisions for randomly taken application components and their corresponding data inputs. Table 1 shows the configuration of our simulation. First, we decide particular values for all the parameters and then we start our simulations by choosing one parameter to be varied each time to analyze its effects on the execution cost, keeping the rest of the parameters constant. We show the varying parameter as '*' in the table, while assigning the constant parameters their corresponding values. For instance, in experiment depicted by first row we vary the bandwidth of available network (β) and denote it as '*' in corresponding row of the table. Note that β, δ_{mob} and δ_{cloud} shown in the table are normalized values, whereas ψ and Θ values in table denote the multiplying factor applied to the randomly generated values as stated above.

b.) Results and Analysis

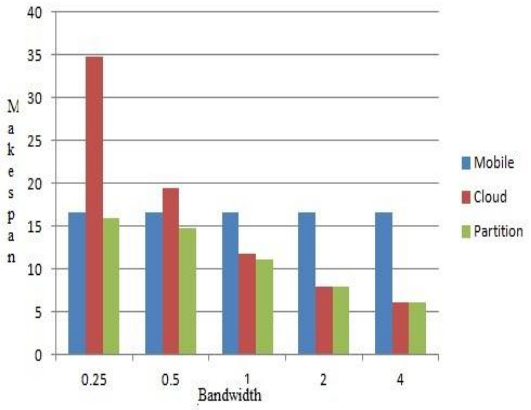
Using table 1, we executed a number of experiments by varying one the parameters each time and the results of the experiments can be seen in fig 2(a)-(f). Below is the analysis about the effects of those parameters on the execution cost of application. Most of the analysis circles around the equation (2) and (3), i.e., the cost of execution at mobile comprises of only computation cost, but execution on cloud requires both computation as well as communication

Fig. 2(a) shows the influence of network parameter β on the execution. It is interesting to find that our approach produces best results among all three approaches throughout the considered range of bandwidth. In low bandwidth conditions, the results of all Cloud execution approach depict significant communication overheads. In such conditions, our approach tends to keep most of the components on mobile device itself and hence, closely follows the all execution on mobile approach. As the bandwidth starts increasing, the communication overhead ease up and more components start getting offloaded to cloud. As the bandwidth becomes abundant, the constraint shifts from bandwidth to cloud resources and the approach ends up following the all Cloud execution approach.

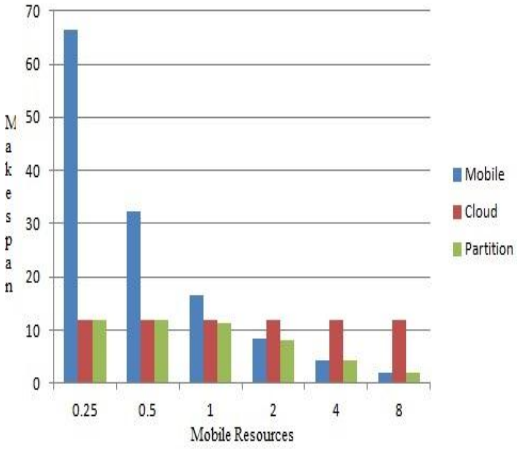
In fig. 2(b), initially the cloud resources become the constraint and the proposed approach offload most of the components on cloud platform to get better performance of to achieve speedup. With the increase in mobile resources, the local execution approach starts dominating and lesser components are offloaded. The proposed approach finally ends up executing all components on mobile.

Table 1: Configuration used in experiments

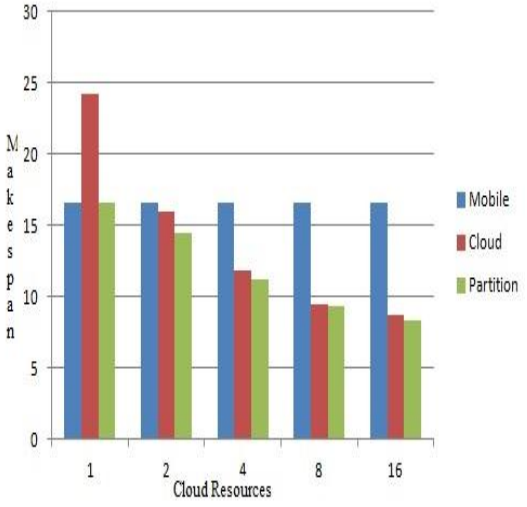
B	**δ_{mob}**	**δ_{cloud}**	**Ψ**	**Θ**
*	1	4	1	1
1	*	4	1	1
1	1	*	1	1
1	1	4	*	1
1	1	4	1	*



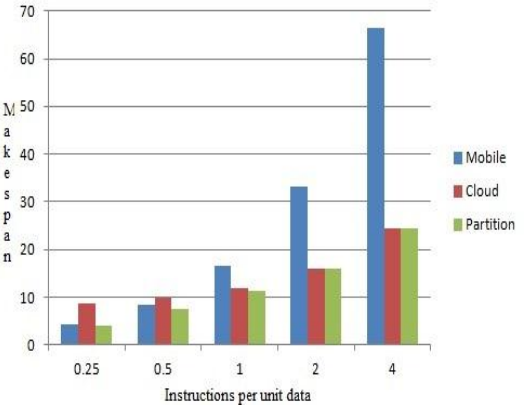
(a) Effects of available bandwidth



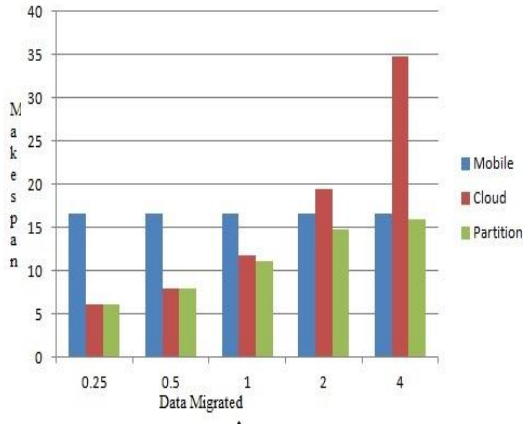
(b) Effects of available mobile resources



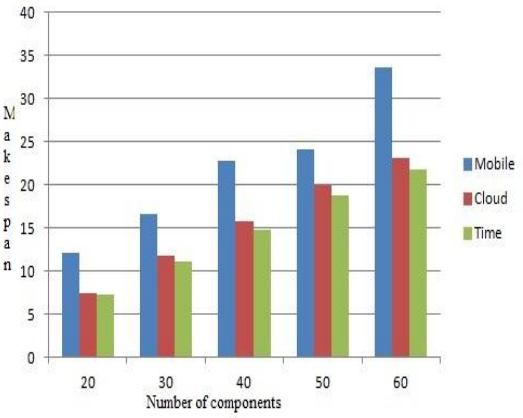
(c) Effects of available cloud resources



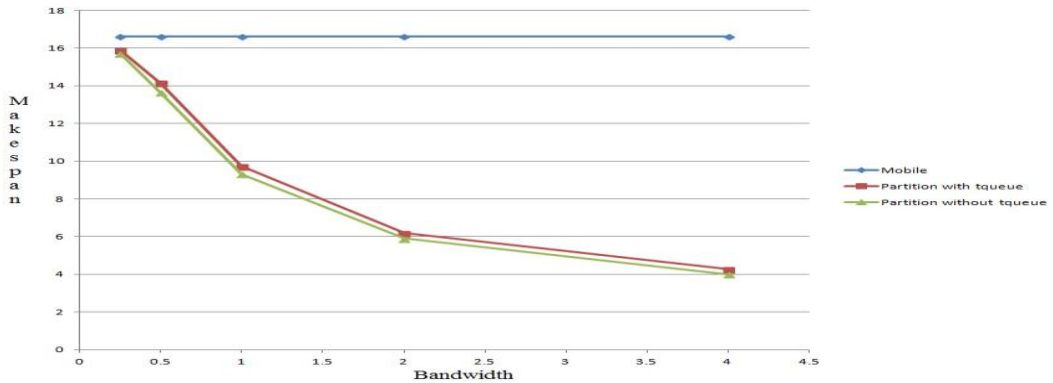
(d) Effects of size of application's components



(e) Effects of data needs to be offloaded per component



(f) Effects of number of components in application.



(g) Partition with waiting time Vs Partition without waiting time

Fig. 2: Simulated Results

The effect of varying cloud resources can be seen in fig. 2(c). With low resources available on cloud, the proposed approach decides not to offload and run most of the components on mobile device itself. As the cloud resources are increased, the cost of execution on cloud resources starts reducing. The reduction in cost of execution on cloud turns the offloading decisions in the favour of cloud, and the approach starts acting similar to all Cloud execution approach.

Interesting trends are seen in fig. 2(d), when the instructions per unit data in the components are varied. With the lesser instructions to be executed, the scenario tends to favour mobile device. This is because lesser number of instructions to be executed as compared to the corresponding data offloading needs. So, the cost of computation gets much smaller than the cost of communication (data offloading) and hence, all the execution is carried on mobile device. With increase in instructions per unit data, the communication versus computation cost ratio turns in favour of communication as the data to be offloaded is lesser as compared to instructions to be executed. Hence, the approach tends to start offloading more and more data and starts following the all-cloud approach.

Fig. 2(e) shows the effect of varying the data to be migrated or offloaded with respect to the components. As the data is low initially, the communication cost is very low and computation cost at cloud side is already low because of much better resources. So, more execution is done on cloud. As the data to be offloaded increases, communication cost becomes a bottleneck and execution on mobile device becomes more cost effective even with lesser resources than cloud.

Varying the number of components in which the application is divided, we see in fig. 2(f) that the proposed approach performs the best among all the three approaches in all scenarios.

In fig. 2(g), we study the difference between the nature of partitioning by considering the waiting time and without considering the waiting time. As consideration of waiting time is expected to result in increase of total makespan of the application, the graph which considers the waiting time hovers slightly above the graph not considering the waiting time. Even though the results are not better but are comparatively closer to the real world.

6. Conclusion & Future Work

Our work presents an approach to make mobile computing to work collaboratively with cloud computing and shows that the collaborative approach proves to be better. Although, the SLA negotiated waiting time poses as a hindrance to the mobile cloud computing but still it is a factor which cannot be ignored.

The simulation results show that offloading approach keeps the cost low in every scenario by exploiting the resources available at mobile side as well as cloud side and minimizes the makespan by 37% on average without considering the waiting time and 34% on average by considering the waiting time. But, as the waiting time is an important factor and cannot be ignored, these results are relatively closer to the real world. We have assumed the whole system to be secure and reliable. So, extra overheads due to security were considered as negligible. But in real world, it cannot be so. So in future work, we would like to incorporate extra overheads due to security and privacy.

References

[1] M. Satyanarayanan, P. Bahl, R. C´aceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," IEEE Pervasive Computing, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", 2009, [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing" [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[4] Shamim Hossain, "ThoughtsOnCloud Cloud Computing conversations led by IBMers", [Online]. Available: http://thoughtsoncloud.com/2013/06/mobile-cloud-computing/

[5] L. Yang, J. Cao, S. Tang, Tao Li, Alvin T. S. Chan, "A framework for Partitioning and Execution of Data Stream Application in Mobile Cloud Computing." IEEE Fifth International Conference on Cloud Computing, 2012.

[6] E. E. Marinelli. Hyrax: Cloud Computing on Mobile Devices using MapReduce. In Master Thesis, Carnegie Mellon Universtiy, 2009.

[7] Gonzalo Huerta-Canepa, Dongman Lee. A Virtual Cloud Computing Provider for Mobile Devices. In Proc. ACM MCS'10, pages 3756–3761. ACM Press, 2010.

[8] D. Kovachev, Tian Yu and Ralf Klamma. "Adaptive Computation Offloading from Mobile Devices into the Cloud." 10th IEEE International Symposium on Parallel and Distributed Processing with Applications, 2012.

[9] Niroshinie Fernando, Seng W. Loke, Wenny Rahayu, "Dynamic Mobile Cloud Computing: Ad Hoc and Opportunistic Job Sharing", Fourth IEEE International Conference on Utility and Cloud Computing, Victoria, NSW: December 2011.

[10] Tatsuya Mori, Makoto Nakashima, and Tetsuro Ito, "A Sophisticated Ad Hoc Cloud Computing Environment Built by the Migration of a Server to Facilitate Distributed Collaboration", in 26th International Conference on Advanced Information Networking and Applications Workshops, pages 1196-1202, Fukuoka: March 2012.

[11] R. F. Lopes, and F. J. D. S. E. Silva. "Migration Transparency in a Mobile Agent Based Computational Grid." In Proc. Of the 5th WSEAS Intl. Conf. on Simulation, Modelling and Optimization, pp. 31-36, Greece, August 17-19, 2005.

[12] E.Truyen, B.Robben, B.Vamhaute, T.Coninx ,W.Joosen, and P. Verbaeten, "Portable Support for Transparent Thread Migration in Java." In Proceedings of 2nd International Symposium on Agent Systems and Applications and 4th International Symposium on Mobile Agents 2000, Zurich, Switzerland, Sept. 13-15, 2000.

[13] T.Sakamoto, T. Sekiguchi, and A. Yonezawa, "Bytecode transformation for portable thread migration in java," In proceedings of 2nd international Symposium on Mobile Agents, Zurich, Switzerland, Sept. 13-15, 2000.

[14] Ricky K.K. Ma, Cho-Li Wang, "Lightweight Application-level Task Migration for Mobile Cloud Computing." In Proceedings of 26th IEEE International Conference on Advanced Information Networking and Applications, 2012.

[15] B.-G. Chun and P. Maniatis, "Augmented Smartphone Applications Through Clone Cloud Execution," in Proceedings of the 12th Workshop on Hot Topics in Operating Systems (HotOS XII). Monte Verita, Switzerland: USENIX, 2009.

[16] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. "CloneCloud: Elastic execution between mobile device and cloud," In Proceedings of EuroSys 2011.

Mr. Gaurav is a Research Scholar pursuing his M. Tech., Computer Science and Engineering II year from Deenbandhu Chhotu Ram University of Science and Technology, Haryana, India. He has completed his B. Tech. degree in Computer Science and Engineering from Maharshi Dayanand University, Rohtak (Haryana).

Mr. Nitesh Kaushik is a Research Scholar pursuing his M. Tech., Computer Science and Engineering II year from Deenbandhu Chhotu Ram University of Science and Technology, Haryana, India. He has completed his B. Tech. degree in Computer Science and Engineering from Maharshi Dayanand University, Rohtak (Haryana).

Mr. Jitender. Bhardwaj is working as Assistant Professor in Deenbandhu Chhotu Ram University of Science and Technology, and also pursuing his Ph. D., from Deenbandhu Chhotu Ram University of Science and Technology, Haryana, India. He has completed his B. Tech. and M. Tech. degree in the past and working as Assistant Professor for last few years.