CloneCloud in Mobile Cloud Computing

Zulfiqar A. Memon[1,*], Javed Ahmed[2,+], Jawaid A. Siddiqui[2,+]
[1]Department of Computer Science National University of Computer and Emerging Sciences, Karachi, Pakistan
[2]Department of Computer Science, Sukkur IBA University, Airport Road, Sukkur, Sindh, Pakistan

Summary
Due to the advancements in computation technolo- gies, apparent in small, faster, and parallel processing capabilities both in hardware and software, there has been a paradigm shift to ubiquitous computing. Most of the computing platforms exist in the form of mobile computing, where portability and round- the-clock accessibility is the main advantage. Mobile Cloud Computing (MCC) helps bridging Cloud Computing seamlessly into a mobile environment to elastically enhance resource utilization in the on-demand requirements for the mobile users and mobile applications providers. The main goal is the provision of a better experience for mobile users by alleviating problems concerning storage, computational power, and battery. This removal of resource limitation occurs in the form of Task offloading. Cloud has the capability to compute faster using the numerous resources available, more storage performance and a constant supply of power compared to mobile devices, hence "heavy-lifting" the cloud is the optimal approach to mitigate resource limitations. Many models have been proposed which take certain parameters into consideration for offloading such as the speed of the cloud versus the speed of mobile devices, and the bandwidth available for transfer between the cloud and the mobile device etc. More- over, before offloading decisions need to be made on what are "offloadable " elements of the code. This annotation is done either manually by the developers of the applications or automatically through appropriate partitioning tools. CloneCloud, which is the focus of this paper, helps in automatically marking the potential "offloadable " blocks in bytecode in a static fashion and then at runtime determines the optimal offloading. CloneCloud uses virtual machine migration (VMM) to transfer mobile application blocks to a cloud server using either 3G or Wifi. This paper elaborates on the framework details of the CloneCloud, the shortcomings found in this model and our proposed solution.

Keywords
mobile cloud computing, clonecloud, remote procedure calls, virtual machine, offloading.

1. Introduction

Relatively cloud computing is defined as the service which provide the facilities to store and share resources, information, and software to other computer and portable devices. This service provided over the Internet. The basic purpose is to facilitate the users to their resources without having knowledge of physical location and provide transparency from requesting resources to getting back their response is the major deter- mination of cloud computing.

As Mobile cloud computing (MCC) is just belong to cloud computing. It refers as the simple infrastructure where data processing and storage are kept on the cloud. It provides a broader range of user to facilitate this service. The main objective of MCC is to deliver better experience to the users who have limited data storage, processing, and battery consumption in their mobile devices [1]. It provides ease to mobile application because processing and storage of data are moved to the cloud. Now the data storage and processing are controlled by powerful computing and accessed over the wireless network. This reduces the need of users to have wide CPUs speed and memory to run dense mobile applications as all the complicated processing are now done in the clouds. Further, user can access their resources and applications from anywhere as the cloud is distributed over the world at different locations. For instance, (Google, Amazon, etc) having their datacenters at different geographical area. Private and local clouds have limited scalability but high performance in terms of caching, battery consumption, delays, and more important privacy on the user's devices.

In this paper, we provide a survey of the challenges that exist in the mobile computing and cloud computing architecture, its scope, and the future enhancements. We also focus on CloneCloud model that provides an avenue to bridge the gap between mobile devices and the efficient processing on the cloud using cloning techniques. We elaborate on the structure of CloneCloud that aids in dynamic partitioning of mobile application in "offloading" seamlessly into the cloud. We also show where limitation CloneCloud lies and how we can manage to optimize it for improving QoS of applications.

The rest of the paper is organized as follows. Section II gives an overview of the MCC and its challenges. In Section III CloneCloud has been described. We then elaborate on the design of CloneCloud's partitioning components in Section IV. Section V delineates the Clone Cloud's limitations and the proposed improvements. Finally Section VI concludes the paper.

Manuscript received August 5, 2017
Manuscript revised August 20, 2017

2. Mobile Cloud Computing

As the need of mobile applications availability are increases. The cloud is distributed to provide the processing of applications and accessibility at any time. However, in order to achieve this, many issues are occurring. Integration between MCC and mobile networks are also facing many technical challenges. To approaches are used for system architecture. Firstly Client-Server approach, which used Remote Procedure Call mechanisms for inter-process communication through machines, networks and address space. Secondly Virtualization framework which is a core mechanism for cloud computing. Figure 1 shows the ecosystem of Mobile Cloud Computing.

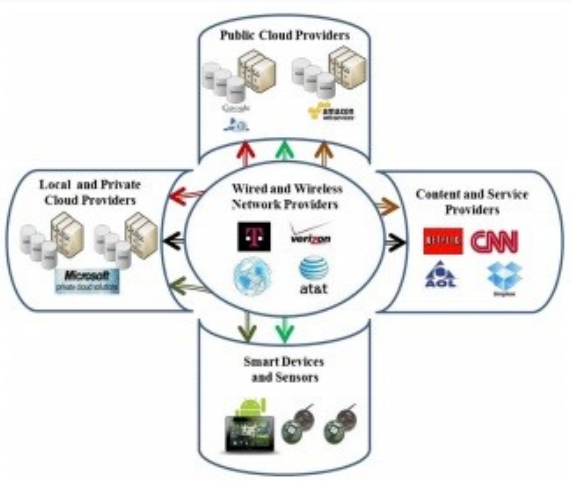

Fig. 1. Ecosystem of MCC including its important components.

To increase battery lifetime and makes the performance better of applications, offloading is the main mechanism of MCC. But there several issues pertaining to dynamic and efficient offloading under different situations.

A. Offloading in the static environment

As offloading is a main feature of MCC for improving performance. But when we examine in static environment. It has major issues. As Research shows in [2], offloading is not always become efficient to save battery life. Offloading sometime might consumes more energy for code compilation than in local processing. It has also been observed that offload- ing consumes more energy in small size code. For instance, experiment evaluated that when a code after compilation have
500KB in size. For communication offloading consumes only
5% of energy in it and local processing consumes 10% of the device battery. In this way offloading saved 50% of the device energy. But when offloading compute small size of code likes
200KB. It consumes 30% of energy for computation which is more than local processing computation.
It is major problem to decide whether to offload or not and which portion of the codes offload to improve the efficiency for mobile applications. It is also considered that different wireless devices consumes different amount of energy and different data transfer rates while doing offloading.
To solve this problem, a solution was suggested [3]. A program estimated and partitioning the code on the basis of energy consumption before program execution. The best way for offloading the program and code partitioning is made on runtime with dynamism. This will be decided on runtime and there is always a trade-off between the cost of computation and communication.
As computation cost is based on time and communication cost is based on data transmitted size and network bandwidth.
This computation and communication cost trade-off change in runtime execution.
Many solutions for portioning the program before execution are proposed. In the paper by Li et al. [4], another portioning scheme presented for offloading. This scheme was planned for offloading computational mobile tasks. It is based on the data sharing information and time at procedure call level. A branch and bound algorithm is applied on cost graph. The main purpose of this scheme is to minimize the communication and computation cost of total energy consumption. The main idea of this scheme is consider an approximate solution for partitioning the program. This scheme only work for partition procedure call tasks.
Another solution was presented [5], which take decisions for the components of java programs to be offloaded. The program first divided the program and then decided which part of the java program is offloaded on the basis of size of functions. Then calculate the computation cost. After that local and remote execution time is compare by this approach. The cost is also based on wireless transmission for the execution time.
Hunt and Scott [6] proposed Coign, which is an auto- matic system of distribution partitioning. It transforms in to distributed applications and not accessing the codes. The approach constructs a graph to search the best possible distri- bution. This distribution is based on scenario profiling. It uses lift-to-front minimum-cut graph-cutting algorithm for finding the least communication time.
Many approaches uses the size of data, its network bandwidth and execution time for finding the best solution of partitioning the program before offloading. It is very difficult to find the precise execution time for improving performance by offloading because of computation and communication time changes in dynamically runtime. Incorrect execution time may result of inefficient performance of applications.
A more efficient approach is presented by [7], it do not require any execution time for computation of instance.

Online statistics are examining for computation time and find out the optimal timeout. If the computation of programs are not completed after timeout. Then the program will be offloaded. This approach only evaluates the inaccuracy of computation time. It is proved by experiments that the approach also save
17% more energy than other approaches.

B. Offloading in the dynamic environment

The environment changes dynamically may also causes many issues. For instance, a data may be lost on the server when it is response to the client or the data convey may not reach to the destination. Table I shows the suitable offloading techniques relative to different environment.
In the paper by Ou et al. [9], investigate the offloading systems performance working in wireless communication. To evaluate the efficiency of offloading, researchers consider into three situation of execution of an applications. The cases are:

1) When the mobile application works fine locally with offloading.

Table 1. Common mobile computing environmental factors [8].

Changes	Priority level	Description
Client side power level	1	Power can be divided into sufficient and insufficient power levels, which will depend on the particular situation.
Connection status	2	The connection status can be faded, disconnected from the mobile network, or re-connected to the mobile network
Bandwidth	3	The bandwidth varies from time to time, and depends on several factors, such as the network traffic condition, etc.

2) When the application runs in offloading environment without failures.
3) When the application operating on offloading with presence of failure recoveries.

In the third case, when failures occur, the operation of application re-offloaded. The portion in which the failure occurs will be re-offload. This improves the execution time. But there are some limitations of this approach. The mobile connection is a wireless ad hoc local area network. Disconnecting of wireless connection during offloading execution will also result in failure.
Another research considers general behavior of environmental changes [10]. They also provide solution of these changes for offloading. For instance, disconnection at the stage of program execution in case of connection status, the server maintains the execution knowledge after checking the connection status of the client for the running tasks. When the connection recovered for the running tasks, server sends the response to the client. In case of unavailability the server waits for the predefined time period and then deleted. One major drawback of this approach is that it only provides general solution. The approach does not define how to dynamically partitioning the application.

3. CloneCloud

CloneCloud enhances mobile applications by off-loading part of the server from mobile devices on to clone's devices. It is designed to provide a service platform for mobile devices. This service worked for generic mobile device processing. As shown in Figure 2, the system transforms the finest network connection of the relative mobile device, application calculated patterns and cloud from single execution of a mobile device in to distributed execution.
The perception of this type of system lies in following: The execution of the program in the cloud is relatively more secure, faster and efficient than the execution on the mobile devices. The process paying the cost and may be worth it for sending the execution code from the mobile to the cloud and vice versa. The computation is required to find the performance metric

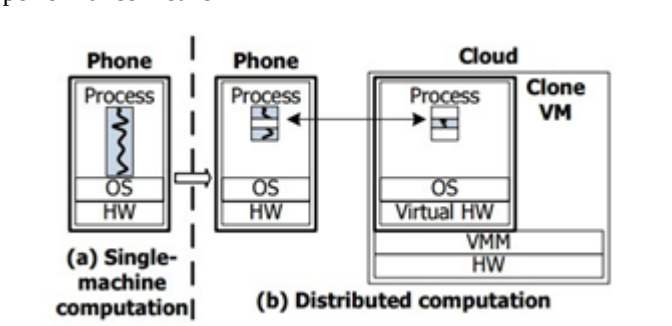

Fig. 2. System Model of CloneCloud.

Between the existing application and the cloud partitioned ap- plication. Cloud late-binds the partitioning if the performance metric of the cloud partitioned code is better. The decision of partitioning is finer grained. It's partitioned different amount of original application on the cloud. Additionally the partitioning not only impacts the application but also impact to the overall performance of the execution. It impacts on the network connection, execution workload, and CPU speed of mobile devices and cloud devices.
The basic aim of Clone Cloud is to take decision about where to run application, and which is more flexible. Another important goal of clonecloud is to take the quality decisions itself without any programmer effort on partitioning the applications. As programmers are also not eager to do hand code job for all types of scenarios a user can face. But also we can imagine the fact that unexpectedly the automatic partitioning do not optimized the application as the experienced programmer can do this hand code. Moreover, the applications on store which gain more popularity having very low motivation to optimize its applications for effective performance and to run on different architecture and networks, consume less battery lives. So, as a result clonecloud goal is to partitioning the applications flawlessly without need of the source code.

In this paper, the mechanism applies on application layer.

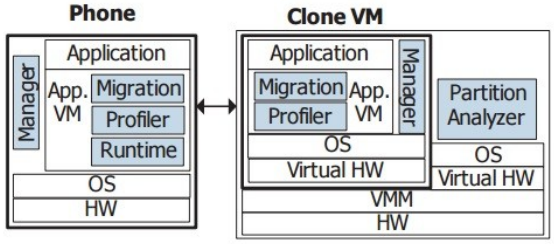

Fig. 3. Prototype Architecture of CloneCloud.

Application layers Virtual Machines (VM) are extensively used on mobile platform. As VM is simpler to migrate and manipulate applications on different sets of architectures. The clone cloud prototype has shown all the design goals on Figure 3. It rewrites the actual application, after modify the application; automatically threads migrate from mobile to cloud devices. The remaining functionality works on mobile devices. It only gets blocks if it required accessing the mi- grated partition code. This is showing opportunistic but also a traditional concurrency. The thread which migrated on cloud, accessing the cloud devices such as accessing the fast CPU, storage, network connections. After processing on the cloud devices the modified state return back to the original process. The decision is taken by partitioned component whether to migrate on the cloud or not. The partitioned components have static analysis on the application which define the constraints and also have dynamic profiling which evaluates the cost for migrating the execution thread. To optimize the calculated total execution time, a mathematical optimize is used. It aims is to optimize the energy consumption on mobile devices and the total time taken by the process.

The model for efficiently use of ambient resources in not new. As Balan et al. [11], also researched for partitioning and executing the application. Further clone cloud is built on exist- ing resources. But it enhance in a novel way. Satyanarayanan et al. [12], research about the clone cloud migration at thread level granularity. It also migrates the code on the cloud except those which require on mobile devices. For example camera and organizing the user interface. Aridor et al. [13], and Zhu et al. [14] work shows that the clone cloud allow migrating operation to execution on mobile and cloud devices. It not only accessing raw resources of cloud devices (i.e. CPU power) but also access specialized resources when fundamental library and OS are try to implement them. MAUI [15], also examine the total cost by static analyzer and dynamic profiler with the help of programmer. The scheme also used optimizer to solve the partitioning issues. Beside these researches, this model can accomplish the performance for mobile applications 20x speed up and also 20x less energy consumption

3.1 Partitioning

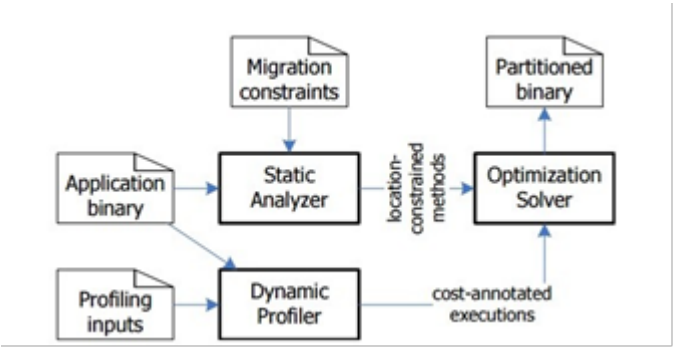

Fig. 4. Workflow Diagram for Partitioning Analysis.

The main aim of partitioning is to decide which part of the mobile application is partition and migrate on the cloud and also decide which part retain on the mobile device for execution. The mechanism of partitioning is made offline on clone cloud. Those applications which have objective to target on the application VM stage may also be partition. The result of partitioning is partition the application part where the execution takes place and migrates from mobile to clone cloud. There is no need a programmer to write and explain some standard rules for it. And there is no need for having source code. Partitioning mechanism works for following parameters: i.e. network connections, energy consumption and CPU performance. It took these parameters and optimizes the total energy an time taken by the mobile device. The mechanism may be work multiple times for different function and different conditions of execution. Also at runtime, the distributed execution selects a partition from the database. It implements the partition using fast modification. In figure 4, the conceptual workflow is shown. The partition made via static analysis of program and dynamic profiling.

A.Static Analyzer

The mechanism of partitioning using static analysis for evaluating the legal choices of partition the application. The choice of placing migrating and reintegrating in the code. According to the rule, the points of migration ad re-integration may be place anywhere in the code. But the process also reduces the legal choices and makes the optimization issues tractable. Further, the mechanism also restricts the migration from entry point and re-integration in the code to the exits point. This paper also makes two restrictions. Firstly the migration can only take place on the boundaries of application not on the core system library which also provides easiness on the implementation at runtime. Secondly the migration can take at VM layer, not on the native method boundaries. However this approach is not allowing native methods for migration but it also allowing the migrated methods to call up the native methods.

In Fig. 5, significant parts of the program are shown. Its static control flow and partitioned graph are shown by an example. It shows a class C which has three methods. Firstly, function a() calls function b(), which carry out expensive processing. The static graph estimate the overall control flow. Exact flow is not determined because the program accomplishment is not determined. The estimation of the program is traditional because the program execution follows many different paths than the path exists in the graph but the communication does not hold. In figure 5(b), the static flow graph represent the method entry exist nodes. It is marked as <class name>.<method name>.<entry| exist>. In figure 5(c), partitioned graph is represented, the body of method c() works on the clone and the other part of the program runs on the mobile device. As method c() is not a native method but it can be call up the native methods.

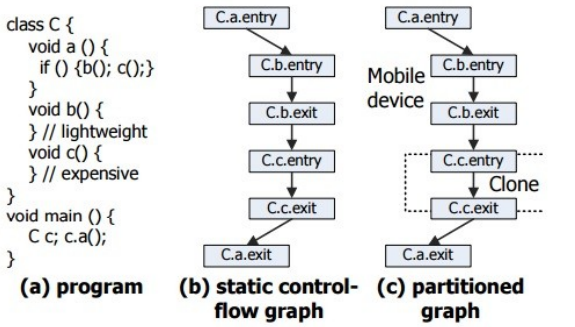

Fig. 5. Static Analyzer program, control-flow graph, and a partition.

1) Constraints: The three following legal properties are defined which require by the migration process. The following properties explain the static analysis which achieves these constraints.

Property 1: The methods that uses specific resources must be retain on a machine.

The property explain that if a method accessing the mobile features such as sensor, GPS location service and camera in a machine then those method should retain and execute on the mobile device. This property mainly concerned about the native and main methods of the program. Such methods are declared as (M) for mobile device. These methods manually recognized in the VM's API. For instance, methods of VM API clearly refer to the camera. This will not repeat for each application an it is handle by the application platform once. The main methods are also marked in the V_M.

Property 2: Methods that have share native state must retain on a same machine.

Applications have native methods that share the same state below the VM. It share and access the native state. As this mechanism does not migrate the native methods on the clone. Such native method must be placed on the machine. For instance, a class of image processing has initialize, defect and fetch result methods that want to access native state, so these methods must be placed on the machine. Native state should be supposed automatically by avoid the burden and it works well. All native are declared by Nat_C in the class C for the set V_{Nat_C}.

Property 3: Prevent nested migration.

The property implies that for one smart phone and clone, no suspends and resumes should be nested. If a program is migrated once at the entry point, there should be no suspended until a resume occurs. The process of re-integration and migration should be executed alternatively. The static analysis creates a static control-flow graph to implement this property. It captures the caller method. As shown in figure 5, if method a() is the partitioning entry points, then the entry points not shift to b() or c().

B. Dynamic Profiler

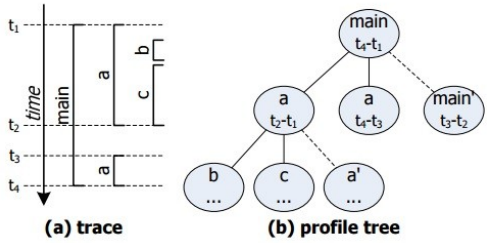

Fig. 6. An example of an execution trace and it's corresponding profile tree.

The function of profile is to build a cost model for the program execution. This cost model used energy consumption and execution time for the mobile devices. Multiple executions invoked by the profile for the applications. Each execution selected randomly input data and execute on clone cloud and mobile devices. The set 'S' output by the profiler of execution and also profiler tree T and T' created.

The tree shown in Figure 6 represents a compressed execution process in one platform. The node on tree had shown each method invocation. The root of the tree is the starting point of the function such as main. This method is user-defined method in the application. Further calls for methods are presented as edges in the tree. These calls are from the main root parent method to the child methods. The order of method is not significant. Every node particularly explains the cost metric representation in the tree. This cost metric is defines execution time. Every method invocation from the parent to its children, non leaves also have a leaf child, which is called residual node. These nodes represent cost of running execution body. It excludes the cost of those methods which is called by it. So each node shows the child node invocation state size and also the end invocation state size. This amount of output is needed for the migratory to view and transmit in different directions.

3.2 CloneCloud Optimization

CloneCloud has seamlessly allowed single-machine execution on a mobile device into a distributed computation using cloud cloning technology. It has achieved this via integrating thread granularity of application code migration. The mobile application can operate in a halt-free manner both on the local device as well as on the clone. This has enabled optimum utilization of power in a resource-constrained mobile device and in a high computational environment of the cloud.However, while it allows virtualization of most components of the executable code in the cloud it has a limitation of not permitting virtual access to native resources. In the case where the method of the application requires access to truly local resource such as the camera/GPS on the mobile phone, it makes the right decision to keep that method pinned on the device. Although, in cases where hardware facilities such as the network or the OS component, which are present both on the mobile device as well as on the clone in the cloud, the non-virtualized access can be detrimental to performance. It would be more optimal to allow migration of such methods in conjunction with an RPC-like mechanism to enable access to remote resources.

This virtualization can occur in the similar fashion to other "offloadable" components. An image can be created of the application and sent to the surrogate (on the cloud) for execution [16]. This can then incorporate migration of previously un-virtualized parts which are then synchronized periodically. This will increase efficiency and allow powerful execution using resources unavailable locally on the device. Similarly, the RPC-based approach can be utilized more robustly to execute code reliant on the device's hardware. Dynamically offloading such mechanism which increases parallelism can protect against the bottleneck and thereby bring drastic im- provement to the CloneCloud model.

4. Conclusion

This paper has shed some light into the existing problems in Mobile Cloud Computing. The various technologies which are present to tackle issues relating to efficient offloading mechanisms which truly and cohesively bridge the gap between mobile devices and cloud computing. The architecture of CloneCloud has been elaborated in this paper, and about how it offers an adaptable code partitioning capabilities by taking in runtime parameters based on storage capacity of the device, computational power required and offered at the clone, and the network cost in migrating the code. CloneCloud achieves the challenges of providing basic augmented execution of mobile application using resources of the cloud. As a step further, this paper has also suggested optimization by virtualization of methods that are reliant on local hardware but can be executed efficiently both on the cloud and by cultivating the RPC-based mechanism for local mobile device components.

References

[1] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasub- ramanian, "Mobile cloud computing: A survey, state of art and future directions," Mobile Networks and Applications, vol. 19, no. 2, pp. 133–143, 2014.

[2] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," ACM SIGMOBILE Mobile Computing and Communications Review` vol. 2, no. 1, pp. 19–26, 1998.

[3] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" Computer, vol. 43, no. 4, pp. 51–56, 2010.

[4] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," in Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems. ACM, 2001, pp. 238–246.

[5] G. Chen, B.-T. Kang, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and R. Chandramouli, "Studying energy trade offs in offloading computa- tion/compilation in java-enabled mobile devices," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 9, pp. 795–809, 2004.

[6] G. C. Hunt, M. L. Scott et al., "The coign automatic distributed partitioning system," in OSDI, vol. 99, 1999, pp. 187–200.

[7] C. Xian, Y.-H. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in Parallel and Dis- tributed Systems, 2007 International Conference on, vol. 2. IEEE, 2007, pp. 1–8.

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

[9] S. Ou, K. Yang, A. Liotta, and L. Hu, "Performance analysis of of- floading systems in mobile wireless environments," in Communications,2007. ICC'07. IEEE International Conference on. IEEE, 2007, pp.1821–1826.

[10] M. Tang and J. Cao, "A dynamic mechanism for handling mobile com- puting environmental changes," in Proceedings of the 1st international conference on Scalable information systems. ACM, 2006, p. 7.

[11] R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H.- I. Yang, "The case for cyber foraging," in Proceedings of the 10th workshop on ACM SIGOPS European workshop. ACM, 2002, pp.87–92.

[12] M. Satyanarayanan, M. A. Kozuch, C. J. Helfrich, and D. R. OâAˇ Z´ Hal- laron, "Towards seamless mobility on pervasive hardware," Pervasive and Mobile Computing, vol. 1, no. 2, pp. 157–189, 2005.

[13] Y. Aridor, M. Factor, and A. Teperman, "cjvm: a single system image of a jvm on a cluster," in Parallel Processing, 1999. Proceedings. 1999 International Conference on. IEEE, 1999, pp. 4–11.

[14] W. Zhu, C.-L. Wang, and F. C. Lau, "Jessica2: A distributed java virtual machine with transparent thread migration

support," in Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on. IEEE, 2002, pp. 381–388.
[15] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in Proceedings of the 8th international conference on Mobile systems, applications, and services. ACM, 2010, pp. 49–62.
[16] D. Lima, H. Miranda, and F. Taïani, "A new system model for cloud offloading."

Zulfiqar Memon is an Associate Professor at National University of Computer and Emerging Sciences (FAST – NUCES), Department of Computer Science, Karachi, Sindh, Pakistan. Previously he has worked as Director Research (Office of Research, Innovation and Commercialization (ORIC), at Sukkur IBA. Dr. Zulfiqar has obtained various funded projects from many reputable international and national organizations that includes, Asian Development Bank (ADB), USAID, World Bank, Higher Education Commission (HEC) Pakistan, National ICT R&D Fund Pakistan and many national and international Non-Governmental Organizations (NGO) as well that includes Institute of Rural Management (IRM) & National Rural Support Organization Pakistan (NRSP).

Javed Ahmed is an Assistant Professor at Sukkur IBA University in Computer Science Department. He has vast teaching experience in prestigious higher education institutions of Pakistan which include FAST-NU, IBA Karachi, Sukkur IBA, and ICMAP Karachi. He also has vast research experience in Semantic Web, Privacy & Data Protection, and Multi Agent Systems

Jawaid Ahmed Siddiqui recieved the BCS, MCS, MS(SPM) degrees from SALU in 1998, University of Sindh in 2001 and from NU-FAST Karachi 2009. After working as Lecturer (from 2002) to dec-2010 and as Assistant Professor from 2011 to date in Sukkur IBA University. His research interest in Software Engineering, knowledge Engineering, Intelligent Systems and HCI.