IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 5, May 2016

A Framework for Code Offloading in Wearable Computing

Divya V[1], Suresh Kumar K R[2]

PG Scholar (Software Engg), Dept. of Information Science and Engineering, M S Ramaiah Institute of Technology

(Autonomous Institute Affiliated to VTU) Bangalore, Karnataka, India[1]

Assistant Professor, Dept. of Information Science and Engineering, M S Ramaiah Institute of Technology

(Autonomous Institute Affiliated to VTU) Bangalore, Karnataka, India[2]

Abstract: Wearable Computing provides a latest form of communication between computer and human such as Google glass, iWatch, SmartWatch which can be worn and always be accessible. Wearable devices come with limited computational capability, battery capacity, and storage. There is a need to increase the same and is done using three tier architecture having wearable computing devices, mobile devices and a remote server by using the code offloading technique. Here computational tasks are offloaded from wearable computing devices to local mobile devices or remote server according to the space and computational capability requirement of the application. A strategy to increase the number of tasks which have to be computed on wearable devices with decreased delay is proposed. A demonstration model is developed to show the offloading process. The values are observed to show the delay if code is not offloaded and they are compared to show the improved efficiency of the model.

Keywords: Wearable computing, code offloading, three tier architecture, mobile devices, remote server.

I. INTRODUCTION

Wearable computing devices are the electronic devices which are worn on the body or clothes such as Google glass, SmartWatch, Magic rings, Headbands, Bracelets etc. Application of wearable devices such as reality augmentation, healthcare monitoring and object or gesture recognition require high communication capability and fast processing in an efficient manner with the less usage of energy. The wearable devices are of limited weight and size as it has to be worn on the body and has less processing capability. Hence it is hard to process more complicated applications. To deal with this computation problem, there is a need for optimized solution which makes the wearable devices to work efficiently without any latency. It is motivated by the fact that users who are using it cannot tolerate much delay while using and operating the wearable device.

Considering the size and weight constraints of wearable computing devices, they are provided with low end hardware and powered by batteries with limited capacity. Hence, they can only run some simple applications which requires less computation capability. To improve and support more complex applications with improved energy efficiency, we propose and use a framework with three tier architecture and code offloading capability. In this proposed framework, the first tier is comprised of the wearable devices, the second tier is made up of the mobile devices and the third tier contains the remote servers normally referring to as cloud.

This framework concentrates on designing an optimal resource management for the complex task computation by the wearable devices. Here some codes from wearable computing devices are offloaded to mobile devices and then to the remote servers (or cloud) depending on the resource requirement. This results in processing of multiple complex tasks on wearable devices with less delay and increased efficiency. The different layers of the proposed three tier architecture communicate through different technologies. Bluetooth or Zigbee technologies are used for the communication between wearable devices and mobile devices within short-range of distance. The communication between mobile devices and remote server happens using WiFi or Long Term Evolution (LTE) networks.

II. RELATED WORK

Wearable computing technology provides many opportunities which give rise to the ideation and nurture of people of all fields. In this age of technology, the reliance on the sophisticated computing devices and associated interfaces are present everywhere. This requirement made way for the growth of wearable computing technology.

These are the mobile computers which can assist people in personal activities by aiding and escalating everyday life. In reality, the constraints imposed such as processor power, battery life, display brightness, and network coverage have led to the delay in the global introduction of wearable computing devices. However in the past few years many successful implementations and the continuous effort to decrease the size of computers promise the emergence of more feasible applications. Outsourcing of computations from mobile devices to the remote cloud is a critical technique as explained by several researchers in their work.

 DOI 10.17148/IJIREEICE.2016.4524

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 5, May 2016

To enhance the communication capability [1], data processing [2], and outsource antivirus services [3] from mobile devices to cloud many techniques were used and these techniques lead to high communication cost as they deal with complete code offloading from mobile devices to cloud. Hence partition techniques were emerged to outsource a part of the application to increase the performance with reduced cost.

MAUI [4] provides code offloading in the form of method level for Microsoft .NET applications. This framework is platform specific. This offers offloading code from mobile devices to remote servers. Offload decisions are taken at runtime. It decides which methods should be executed on remote infrastructure.

CloneCloud [5] technique uses a combination of static analysis and dynamic profiling to partition applications automatically at a fine granularity which optimizes execution time and energy use for a target computation in the cloud and communication environment. At runtime, the application partitioning which is dynamic is effected by migrating a thread from the mobile device at a chosen point to the clone in the cloud, executing there for the remainder of the partition, and re-integrating the migrated thread back to the mobile device. Here prototype designed delivers up to 20x speed and 20x energy reduction for the simple applications.

Static Partitioning [6] technique mainly deals with improving battery lifetime where multimedia data are heavily processed when transferred between the server and the handheld. To minimize battery consumption on the handheld device, a tradeoff has been made between the energy consumed by computational processes versus that by data transfer.

Gaussian Linear Algebraic technique [7] is used to offload the code from mobile devices to remote server. It solves a system of linear algebraic equations. In Procedural Call technique [8], tasks are partitioned into server task and client task. Sever tasks are processed in server. Client tasks are processed in mobile device. Computation and data sharing are handled dynamically at the time of procedure calls.

Branch and Bound algorithm [9] technique investigates hardware/software partitioning, a key problem in embedded co-design system. An efficient algorithm is proposed to optimally solve the problem in which the communication overhead is taken into account. The proposed algorithm constructs an efficient branch-and-bound approach to partition the hot path selected by path profiling techniques. The techniques for generation of good initial solution and the efficient lower bound for the feasible solution are customized in branch and bound search.

In Comparison technique [10] local/remote decision logic is incorporated in the proxy of each component that is considered suitable for remote execution. The proxy can be either defined by the programmer at development time or automatically created using profile data when the application is deployed on the server for the client to download. When the interface method of the proxy is invoked, the proxy evaluates and compares the local execution and offloading costs for the invocation. If the method is determined to be executed locally, the proxy also determines the optimization level for the methods of the local implementation of the component.

Abstraction technique [11] partitions a program into the distributed subprograms by producing a program abstractions where all physical memory references are mapped into the references are mapped into the references of abstract memory location. . This scheme partitions an ordinary program into a clientserver distributed program, such that the client code runs on the handheld device and the server code runs on the server. Partition analysis and program transformation guarantee correct distributed execution under all possible execution contexts. A polynomial time algorithm to find the optimal program partition for given program input data is given. An option-clustering approach to handle different program partitions for different program execution options. Experimental results show significant improvement of performance and energy.

ThinkAir [12] is the parallel execution technique in cloud for code offloading. Unlike other techniques, ThinkAir do not offload the entire code and instead concentrates on some part of the code to be offloaded. In the proposed framework, we use ThinkAir technique and adopt it with our architecture with some minimal changes. In our proposed system, we use the computational capability of ThinkAir technique on the wearable devices.

III. THREE TIER ARCHITECTURE

The three tier architecture as shown in Fig 1, the code is offloaded efficiently from wearable device from tier 1 to higher efficient devices in the tier 2 and tier 3.

In the proposed architecture, the wearable application is categorized into set of wearable device tasks and set of non-wearable device tasks based on the computational capacity of the devices in the respective tiers. Wearable device tasks in tier 1 are the ones that are carried out on wearable devices and cannot be offloaded such as sensing or displaying.

These tasks are denoted as non-off loadable tasks i.e., non-o-tasks. Non wearable tasks are the ones that cannot be performed on the wearable devices and needs to be offloaded onto the higher computational devices or onto cloud. These are referred as off loadable tasks, also known as o-tasks. The set of tasks carried out by the mobile devices in tier 2 are the non-o-tasks of tier 2 and the ones that are offloaded to tier 3 are the o-tasks of tier 2.

Tier 1 and tier 2 comprises of both non-o-tasks and o-tasks based on their computational capacity. Assuming the tier 3 to be of with high computational servers/cloud, tier 3 will not offload the code further and hence comprises of only non-o-tasks.

DOI 10.17148/IJIREEICE.2016.4524

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH IN ELECTRICAL, ELECTRONICS, INSTRUMENTATION AND CONTROL ENGINEERING
Vol. 4, Issue 5, May 2016

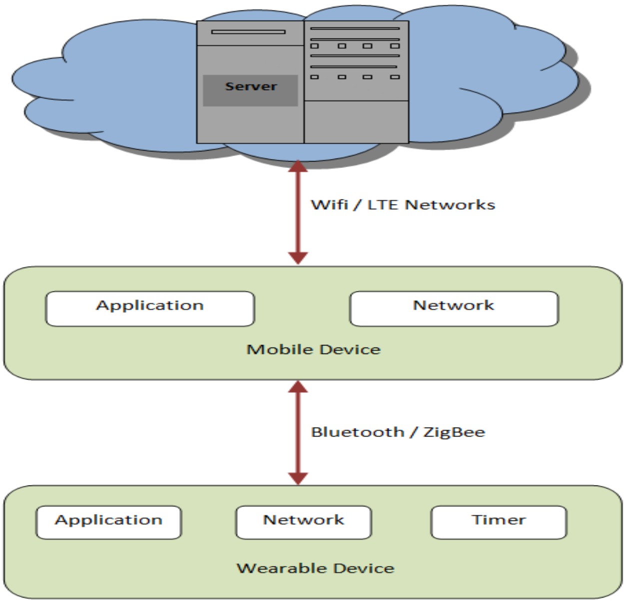

Figure 1: Three tier Architecture

IV. CODE OFFLOADING

Offloading [14] is a process of executing the code on remote servers or on cloud or on the high computational devices which are delegated by wearable devices or mobile devices or low computational devices. In this framework there are wearable devices in tier 1, a smart mobile phone in tier 2 and a high computation server/computer (normally referring to cloud) in tier 3. Both tier 1 and tier 2 constitutes nodes with mobility. Tier 1 and tier 2 communicate using Bluetooth/ZigBee technology while tier 2 and tier 3 communicates using WiFi/LTE network. The o-tasks are offloaded using client server model from wearable devices to mobile device and from mobile device to the server/cloud. When wearable device detects the o-tasks through Bluetooth it offloads the code to the mobile device and mobile device executes the tasks if it does not exceed its capacity. If the o-tasks offloaded from the wearable device to mobile device exceed the capacity of mobile device, then that task is identified as o-tasks in tier 2 and it is offloaded to the server/cloud. There is no further offloading of tasks from tier 3 and contains only non-o-tasks.

V. EXPERIMENTAL SETUP

To demonstrate the proposed three tier architecture framework for code offloading, we take the health monitoring system as an application. The experimental setup is as shown in Fig 2. A heart rate sensor with Bluetooth compatibility as a wearable device in tier 1and is referred as an embedded unit. We have a smart phone in tier 2 referring to as the android unit. In tier 3, we have a web server/cloud.. Due to the size limitation, the heart rate sensor in the embedded unit does not have memory and hence when it detects the heart rate for specified time, it offloads the readings to the mobile device present in the android unit. Mobile devices receives the heart rate and stores the specified readings. Mobile device also has certain limitations. It cannot identify critical situations. Hence from mobile device all the readings are offloaded to remote server in tier 3 having a high computational capability and are monitored by hospital technicians. Once remote server identifies the critical situation, it tracks the location of the patient from the mobile GPS and gives instant message to the emergency/ambulance services.

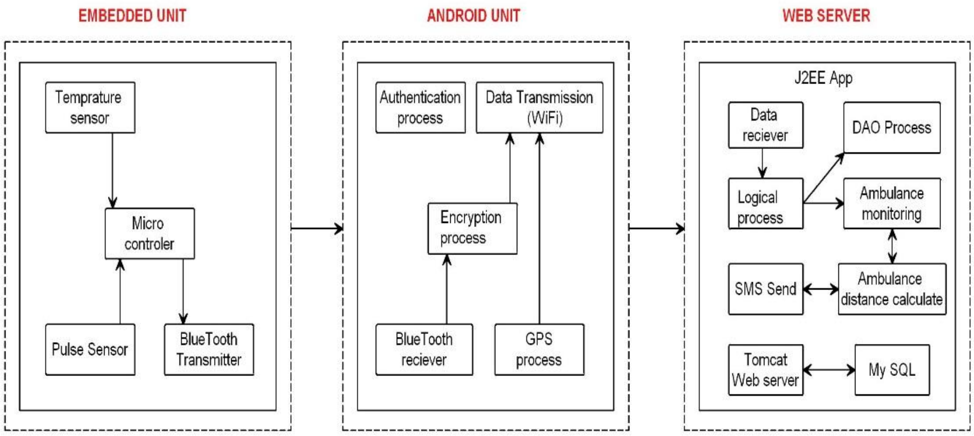

Figure 2: Experimental Setup

VI. CONCLUSION

The majority of the earlier techniques dealt with offloading code directly to the remote server though the task was not so complex and resulted in the wastage of resources in the high computational servers. Here, we use three tier architecture and depending on the computational requirements, the code is offloaded the higher tiers.

The use of mobile device in tier 2 decreases the communication latency. The wearable devices with limited capabilities cannot perform all the tasks and hence need the tasks/code to be offloaded to the high computational devices. The proposed system with the offloading capability is efficient when the wearable device alone cannot perform the tasks within their limited capabilities.

REFERENCES

[1] Luo, Xun. "From augmented reality to augmented computing: a look at cloud-mobile convergence." Ubiquitous Virtual Reality, 2009. ISUVR'09. International Symposium on. IEEE, 2009.
[2] Marinelli, Eugene E. Hyrax: cloud computing on mobile devices using MapReduce. No. CMU-CS-09-164. Carnegie-mellonuniv Pittsburgh PA school of computer science, 2009.
[3] Oberheide, Jon, Kaushik Veeraraghavan, Evan Cooke, Jason Flinn, and FarnamJahanian. "Virtualized in-cloud security services for mobile devices." In Proceedings of the First Workshop on Virtualization in Mobile Computing, pp. 31-35. ACM, 2008.
[4] Cuervo, Eduardo, ArunaBalasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and ParamvirBahl. "MAUI: making smartphones last longer with code offload." In Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 49-62. ACM, 2010.
[5] Chun, Byung-Gon, SunghwanIhm, PetrosManiatis, MayurNaik, and Ashwin Patti. "Clonecloud: elastic execution between mobile device and cloud." InProceedings of the sixth conference on Computer systems, pp. 301-314. ACM, 2011.
[6] Li, Zhiyuan, Cheng Wang, and Rong Xu. "Task allocation for distributed multimedia processing on wirelessly networked handheld devices." InParallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002, Abstracts and CD-ROM, pp. 6-pp. IEEE, 2001.
[7] Rudenko, Alexey, Peter Reiher, Gerald J. Popek, and Geoffrey H. Kuenning. "Saving portable computer battery power through remote process execution."ACM SIGMOBILE Mobile Computing and Communications Review 2, no. 1 (1998): 19-26.
[8] Li, Zhiyuan, Cheng Wang, and Rong 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, pp. 238-246. ACM, 2001.
[9] Jigang, Wu, and SrikanthanThambipillai. "A branch-and-bound algorithm for hardware/software partitioning." In Signal Processing and Information Technology, 2004. Proceedings of the Fourth IEEE International Symposium on, pp. 526-529. IEEE, 2004.
[10] Chen, Guangyu, Byung-Tae Kang, MahmutKandemir, Narayanan Vijaykrishnan, Mary Jane Irwin, and RajarathnamChandramouli. "Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices." Parallel and Distributed Systems, IEEE Transactions on15, no. 9 (2004): 795-809.
[11] Wang, Cheng, and Zhiyuan Li. "A computation offloading scheme on handheld devices." Journal of Parallel and Distributed Computing 64, no. 6 (2004): 740-746.
[12] Kosta, Sokol, AndriusAucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading." In INFOCOM, 2012 Proceedings IEEE, pp. 945-953. IEEE, 2012.
[13] Flores, Huber, Pan Hui, SasuTarkoma, Yong Li, Satish Srirama, and RajkumarBuyya. "Mobile code offloading: from concept to practice and beyond." Communications Magazine, IEEE 53, no. 3 (2015): 80-88.
[14] Cheng, Zixue, Peng Li, Junbo Wang, and Song Guo. "Just-in-time code offloading for wearable computing." Emerging Topics in Computing, IEEE Transactions on 3, no. 1 (2015): 74-83.