I.J. Information Technology and Computer Science, **2017, 9, 45-52**
Published Online September 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2017.09.04

An Experimental Analysis on Performance and Energy Saving in Mobile Cloud Computing

Sindhu K
Department of ISE, BMS College of Engineering, Bangalore, India
E-mail: Sind19@gmail.com

Dr. H.S.Guruprasad
Department of ISE, BMS College of Engineering, Bangalore, India
E-mail: drhsguru@gmail.com

Received: 02 June 2017; Accepted: 10 August 2017; Published: 08 September 2017

Abstract—Mobile Cloud Computing is a combination of mobile, cloud and wireless network where data storage and processing happens outside the mobile device. The storage capacity, processing power and battery life can be improved by moving resource intensive applications onto the cloud. In this paper, the performance of the mobile device is measured by using four different sorting techniques. Two different strategies were used for execution. In the first strategy, the input, execution and the output happens on the mobile device. In the other strategy, the input and output happens on the mobile device while the execution of the sorting techniques is offloaded to the server. The parameters considered for measurement are the execution time and mobile device's energy consumption. The results show that offloading the task to the server reduces the execution time and energy consumption on the mobile device.

Index Terms—Mobile cloud computing, android, mobile, performance, energy consumption, offloading, server.

I. Introduction

With rapid growth of smart phone users, scaling up the capability of smart phones has become a major concern. Mobile devices still lack in resources like low battery power, storage and processing. To overcome these challenges the resource intensive tasks of mobile can be moved to resource rich environment which is external to the mobile. There are many challenges to be addressed in Mobile cloud computing like availability, efficient and dynamic offloading, low bandwidth, heterogeneity, security and data portability.

Mobile cloud computing is a model for transparent elastic augmentation of mobile device's capability using wireless access to cloud storage and computing resources. With the world moving mobile in most of the practical aspects, Mobile cloud computing would facilitate the mobile users with varied services to make the maximum benefit of cloud computing.

Mobile cloud computing enables mobile users to store and process the data in cloud thereby deterring the need to have a powerful device configuration as all resource-intensive computing can be performed in the cloud.

In this paper, a study on the execution time and energy consumption of the mobile device is carried out when the entire program is executed on the mobile device and server separately. Four different sorting techniques were used for the study. It was observed that when processing huge amount of data, it is better to offload the processing task to resource rich environment rather than the mobile, thus execution happens at a faster rate and energy consumption of the mobile device is also reduced.

The remainder of the paper is organized as follows. In Section 2, a discussion on the related work in Mobile cloud computing is done. Section 3 gives a detailed discussion of proposed approach. Results are provided in Section 4 followed by conclusion in Section 5.

II. Related Work

Smart phones are gaining huge popularity because of their support for various applications such as image processing, video processing, natural language processing, gaming and e-commerce. Mobile cloud computing caters to such applications by offloading the computation intensive task on the cloud to enhance the performance of resource constrained mobile devices. The offloading of application component [29] is done mainly in two ways; it can be either static or dynamic. In static offloading, the components of the applications to be offloaded are predetermined. In dynamic offloading, the decision to offload is taken based on various parameters like mobile device resources, bandwidth, latency and availability of cloud resources.

In recent years, many cloud-based mobile application models are proposed by various researchers. The recent frameworks by various researchers have been discussed in the following section.

S. Abolfazli et al [1] has investigated the impact on the performance of smart phone by considering three different strategies for execution native, proximate and distant cloud. The study indicates that the transmission volume and number of intermediatary hops have

considerable impact on resource intensive mobile applications' performance. Two cloud setups were used for the experiment, one located close by to the experimental test bed and other far away from the test bed. The execution time and mobile energy consumption were the parameters considered for the evaluation. M. Shiraz et al [2] proposed a simple active service migration framework using light weight mechanism for computational offloading to cloud. The framework uses coarse granularity level for computational offloading and compares the performance with traditional computation offloading which uses finer granularity. Offline and Online mode of execution of the task were carried out. In the offline mode, the application was executed on mobile and in online mode the intensive components of the application were sent to the cloud. Xia et al [3] tried to improve the applications' execution time on smart phone and save energy for smart phones by offloading the computation of application from mobile phone to cloud. The framework is a semiautomatic offloading system since the application is manually modified to run on cloud and an offloading proxy is used to send and receive information from cloud. The decision to offload or not is based on two parameters; the delay tolerance threshold given by the user and the power consumption to execute the application. S. Abolfazli et al [4] proposed a market oriented architecture to publish, discover and host service on nearby mobile which reduces long WAN latency when distant cloud is used and it creates a business prospect to mobile owners. The framework consists of four modules, the service governor responsible for monitoring and managing all the critical tasks, service developer for development of services to be provided to the requester, mobile host is responsible for executing the service request and would be paid by the service governor and finally the service requester. This architecture is aimed at moving towards greener computing by utilizing the unutilized resources of close by mobile devices. Liu et al [5] investigated a Dynamic Programming Offloading Algorithm for finding which functions of the application program should be offloaded to cloud to save the energy of mobile device and reduce the execution time of the application. The framework consists of a solver module which decides which method of an application should be offloaded to cloud and which shouldn't be offloaded by maintaining a dynamic programming table. The solver module takes the decision based on the input from device profiler and application analyzer which decides the cost to run the method on the mobile device. The solver module also gets the input from constraint analyzer which identifies the methods which cannot be offloaded and other module called network analyzer which gives the transition cost of a method based on characteristics of network and the data to be sent by the method. Xiang et al [6] investigated a new method where multiple mobile application code offloading requests are send in bundles. Usually after offloading the request to the cloud the Wi-Fi interface on the mobile will be in high state for a short duration before switching over to low power state referred as tail time. The authors bundle the requests together to reduce the tail time which in turn reduces the energy consumption. Two online algorithms, one deterministic and other randomized dynamically decide when to grant requests. The work proves a substantial amount of energy savings.

Kaya et al [7] investigated Inversion of Control offloading technique. The objects are created using offloading factory instead of using the new keyword. A remote object and a local proxy for each object are created if the object needs to be offloaded. If the objects from the cloud and mobile needs to communicate then proxies are created at both mobile and cloud end, hence communication happens. If any objects on the mobile are needed at the cloud, proxies of those objects are created at the cloud. Each object is identified by an object Id and the mapping is maintained in a table. Wu et al [8] considers the tradeoff analysis of performance improvement and energy saving by considering the execution time into three intervals namely never offload, tradeoff and always offload. When an application needs to be executed, a decision is made to choose the cloud server based on the priority given to the execution time or energy consumption. The proposed adaptive offloading model can improve performance when application is executed faster on cloud then on mobile device taking into consideration Network bandwidth, amount of data to be transmitted and execution time on server. Fekete et al [9] proposes an offloading technique which optimizes the code in development time i.e. weak part of the software is identified and offloaded for external execution. Based on method score, it is decided that the methods which have highest score are offloaded to the cloud.

Bolla et al [10] proposes an Application State Proxy (ASP) that stops applications on mobile and maintains their presence on any other network device because most of the applications on mobile phones are internet based which generally send and receive messages over Wi-Fi, 3G or 4G data connection. These applications utilize the mobile resources phenomenally. So only when an event occurs i.e. if there is a need for the application, the ASP returns the application control back to mobile. Silva et al [11] presents a framework to enhance the performance of applications in mobile by load partition and offloading. Face recognition application was first executed on mobile phone and using power tutor tool the mobile device energy consumption was measured. Then the application was executed on cloudlets and the virtual machines were increased from one to four and it was seen that there was considerable speedup and energy saving.

Elgendy et al [12] proposes a framework where the decision to offload the application to cloud or not is based on real time decision metrics which are total execution time, energy consumption, memory, remaining battery and security. When the application is first executed on the mobile, the application will also send the jar file which contains all remote services on cloud so that when application is again executed it doesn't transfer anything on cloud. The application when executed for the first time generates profile files containing data on memory used, network bandwidth to be used in subsequent execution

and processing time. Angin et al [13] proposes an autonomous agent based dynamic computation offloading model where the application is partitioned into autonomous agent based modules and sent to the cloud and native application module are executed on mobile statically. The framework consists of an execution manager which resides on mobile and responsible for deciding whether the module of an application should be migrated to cloud or not. The execution manger checks with the directory service manager to get the updated list of available cloud hosts for offloading the application module. High reliability is guaranteed by reestablishment of communication between mobile and cloud incase the connection fails. The execution time is calculated by executing the application on mobile, completely offloading on cloud and offloading only few methods of the application on cloud. The experiment is also carried out by using single thread and multithread executions and varying the instance types of EC2. Troung et al [14] presents an offloading model where a mobile user can use many mobile cloudlets and process parallely all independent tasks. The decision to offload or not is taken by the mobile user based on computation, communication and penalty cost. S. Abolfazli et al [15] proposes a framework by considering a network of adjacent service based mobile cloudlets for remote execution instead of far away clouds. The framework consists of three main building blocks, the Mobile Service consumer which is the mobile device, Mobile service provider are the adjacent mobile cloudlets who needs to register with the Trusted Service Governor, which is the third block responsible for supervising and monitoring augmentation entities. A mobile device which requires the service should contact the service execution handler responsible for listening to the requests. The request to execute the task is given to the mobile cloudlet which is located close to the mobile consumer. The architecture also takes care of reliability by caching the data and reestablishing the connection.

In the proposed work, static offloading technique was used wherein the computation intensive task of sorting the numbers is offloaded on the server. A comparative study is done on the performance of the mobile device by executing four sorting techniques on the mobile device and the server.

III. Proposed Approach

In the proposed approach, four different sorting techniques are considered i.e. Quick sort, Merge sort, Insertion sort and Selection sort. These sorting techniques were experimented on two different execution strategies. In the first execution strategy, the input, execution and the output of the sorting techniques were done on the mobile device. In the second execution strategy, reading the input from the user and the displaying the output were on the mobile end, while the execution of sorting techniques was done on Server. For each sorting technique, the range of input numbers was varied from 5000 to 50000 in steps of 5000. The execution procedure for each input value was repeated 20 times to maintain consistency of the measured data. The total time taken to execute and energy consumed was measured and the average was considered. The time taken to execute was measured for four sorting techniques by varying the input range using both strategies. Energy consumed by the mobile was measured for the four sorting techniques for both execution strategies using power tutor [16] considering only the processor energy usage. The procedure followed for executing the program on the mobile device and on the server, is as given below:

Method 1: Executing the program on the mobile.

The program was written on Android SDK and executed on the mobile by prompting the user to enter the input value. Once the input is obtained from the user, random numbers were generated on the mobile. Then the sorting function was invoked and sorted numbers were displayed on the mobile device. The time taken to execute the program and the energy consumed were measured. The same experiment was repeated 20 times for each input and the average value was considered for comparison. The algorithm is given below:

Algorithm

Step 1: Read the input from the user (N Value).
Step 2: Generate N random numbers and store on mobile.
Step 3: Call the appropriate sorting technique function.
Step 4: Display the result on mobile.
Step 5: Calculate the time taken to execute the program using System.currentTimeMillis() and energy consumed by using Power tutor.

A user interface was designed prompting the user to enter the input range and then display the sorted output on the mobile device. At the start of execution of the program, the input range was requested from the end-user and based on the input range entered; random numbers were generated and stored on mobile device. Based on the sorting technique that need to be executed, the function to perform the sort was invoked and once the sorting was done, the sorted array was displayed on the mobile. The total time taken to execute the program was calculated from the start of entering the input value to the end of displaying the result on the screen. Using Power tutor the energy consumed from the beginning of the program till the end was measured. The same procedure is repeated 20 times for the given input range and then the average of both the time taken to execute and energy consumed is considered. The same experiment is repeated for all input ranges of the selected sorting techniques.

Method 2: Executing the program on the Server.

The user interface was developed on Android SDK and the user was prompted to enter the input value. Once the input is obtained from the user, the value was sent to the server. A program written in PHP on the server generates the random numbers. The sorting function on the server sorts these random numbers. The sorted numbers were

sent from server to the mobile device and displayed. The time taken to execute the program and the energy consumption is measured. The same experiment was repeated 20 times for each input and the average was taken. The algorithm is as given below:

Algorithm (Mobile Part)

Step 1: Internet permission to be added to Android manifest.
Step 2: Read the input from the user (N Value).
Step 3: HttpURLconnection object need to be created to send and receive data over internet.
Step 4: Send the input value to the server by setting the request method to POST and sending the URL address and input N.
Step 5: On receiving the response code HTTP_OK, invoke the server program to perform the sort by passing the URL address of the sort program located on the server.
Step 6: Read the result from the server using InputStreamReader and display on the mobile.

Initially, internet permission needs to be added to the Android manifest xml file because the communication happens between mobile device and the server. Once the input is read, the corresponding server program is invoked and the input value N is sent to the server. After the connection is established, the sorting program is invoked by using HttpURLconnection object. Finally, the sorted output is received from the server and displayed on the mobile device.

Algorithm (Server Part)

Step 1: The input value (N) send from the mobile is obtained and stored on the database of the server.
Step 2: Generate N random numbers and store on the server
Step 3: Call the function to perform appropriate sort.
Step 4: Return the stored data to the mobile for display.

The time taken to execute the program was calculated using System.currentTimeMillis() from the time the input value is entered by the user till the results are displayed on the mobile. (i.e. time taken by the mobile and server). Energy consumed by the mobile is calculated using Power tutor.

On the server-end, two programs were written. First program to read the input value received from the mobile device and store in the database. In the second program, when the sort button is selected on the mobile, the input value was read from the database and the random numbers were generated on the server. The appropriate sorting technique was invoked to sort the random numbers and the results are displayed on the mobile. The total time taken to execute the program was calculated by considering the time taken to read the input and transfer the data from mobile device to the server, time taken to process the task on the server and time taken to transfer the result back from server to mobile and displaying the output. The mobile energy consumed for the same was calculated. The experiment is repeated 20 times for a given input and the average is considered for comparison. All four sorting techniques were executed the same way. Only displaying the output and reading the input value was done on the mobile, the entire random number generation and sorting was done on the server end.

Devices used for Experimental Setup were:

Client - Smart phone Sony Xperia M C1904 featuring dual core processor with 1 GHz speed and 1 GB RAM. Battery Capacity 1750 mAh Li-lon
Server - Intel(R) Core(TM) i3 CPU M 370 @ 2.40 GHz 4GB RAM 64-bit Operating System, x64 based processor.

IV. Results and Discussions

Figure 1 shows the comparison of time taken to execute the Quick sort program on the mobile and the server. Figure 2 shows the comparison of mobile energy consumption when Quick sort program is executed on the mobile and the server. It can be observed that for numbers below 10000, the performance on the mobile device is better compared to execution on the server. As the input increases, the performance of the sorting technique takes a longer duration and consumes more energy when executed on mobile compared to execution on the server.

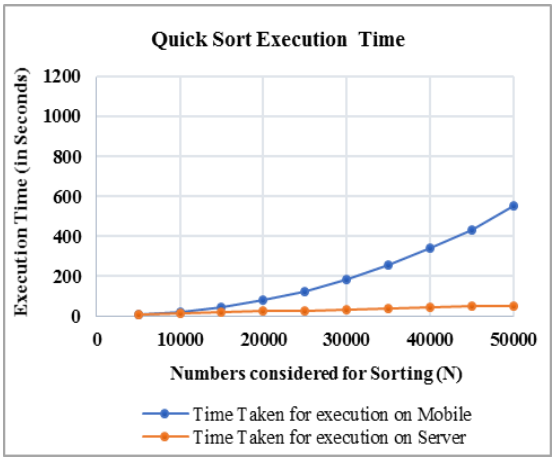

Fig.1. Quick Sort Execution Time

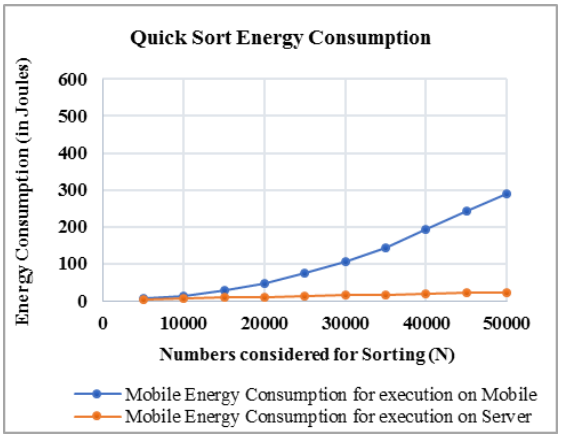

Fig.2. Quick Sort Energy Consumption

Figure 3 shows the comparison of time taken to execute the Merge sort program on the mobile and the server. Figure 4 shows the comparison of mobile energy consumption when Merge sort program is executed on the mobile and the server. For numbers below 10000, the total execution time and energy consumed is less when executed on mobile device. With increase in input size the execution on the server-end yields better result.

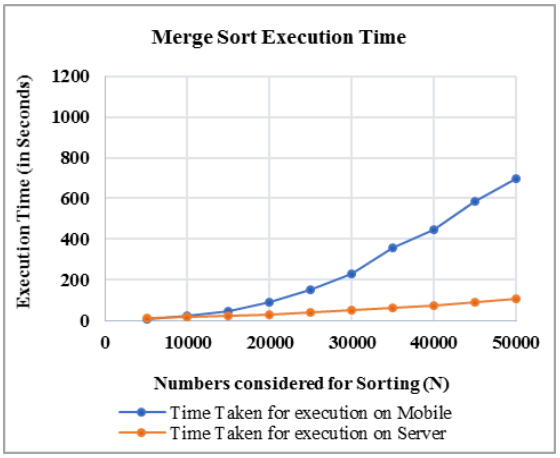

Fig.3. Merge Sort Execution Time

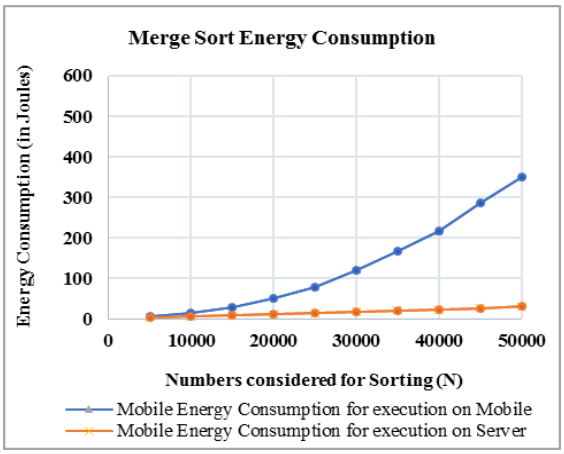

Fig.4. Merge Sort Energy Consumption

Figure 5 shows the comparison of time taken to execute the Insertion sort program on the mobile and the server. Figure 6 shows the comparison of mobile energy consumption when Insertion sort program is executed on the mobile and the server. It is observed when the Insertion sort program is executed on server, time taken to execute and energy consumption is very less compared to execution on mobile.

From the graphs, it can be observed that Insertion sort takes more execution time compared to Quick sort and Merge sort. Insertion sort is faster when the array size is small, but as the size increases the execution time for sorting also increases. When the input is varied from 5000 to 15000 for insertion sort, there is no significant difference in executing on the mobile or server. When the input is increased beyond 15000, increase in performance can be observed when executed on server compared to execution on mobile. Energy consumption of mobile is very less when Insertion sort program is executed on server compared to execution on mobile.

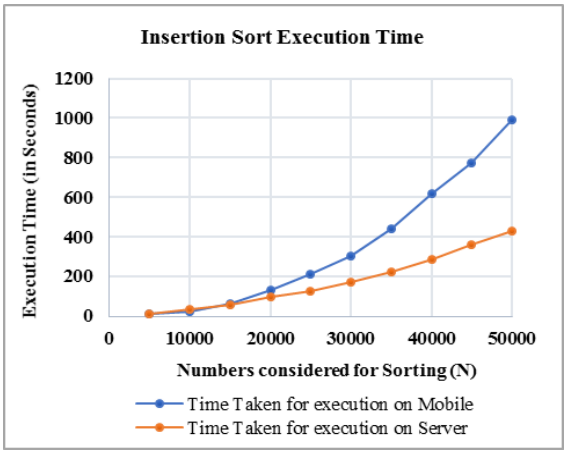

Fig.5. Insertion Sort Execution Time

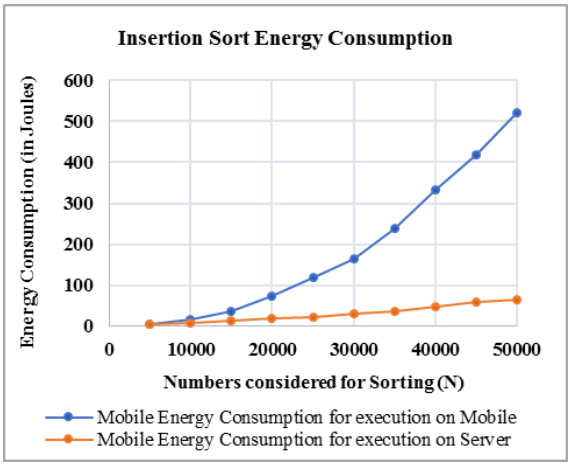

Fig.6. Insertion Sort Energy Consumption

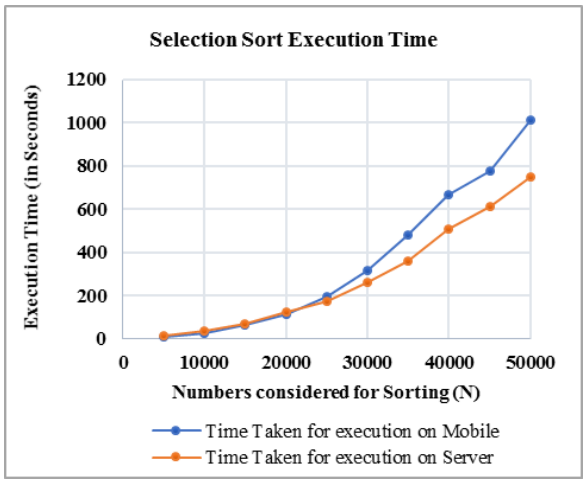

Fig.7. Selection Sort Execution Time

Figure 7 shows the comparison of time taken to execute the Selection sort program on the mobile and the server. Figure 8 shows the comparison of mobile energy consumption when Selection sort program is executed on the mobile and the server. From Figure 7 it can be observed that there is no much significant difference in

time taken to execute when selection sort is executed on mobile or server for numbers below 25,000.

As the input data size increases, the time taken to execute the Selection sort program on server is faster than execution on mobile. The mobile energy consumption, when the Selection sort program was executed on the mobile is less when compared to execution on the server for numbers below 5000. As the input size increases, the mobile energy consumption when executed on mobile device increases gradually when compared to execution on server.

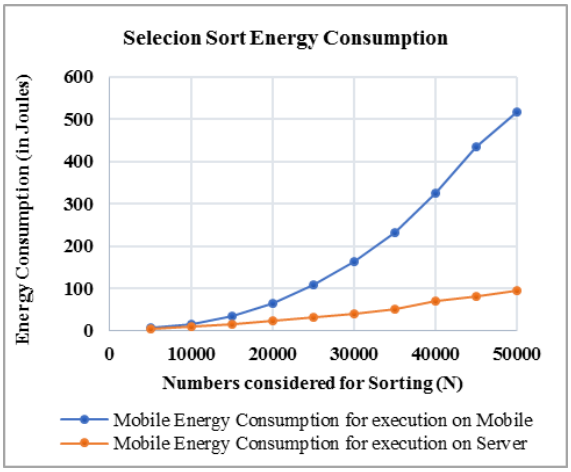

Fig.8. Selection Sort Energy Consumption

Figure 9 gives the comparison of execution time of all the four sorting techniques when executed on the mobile. Figure 10 gives the comparison of execution time of all the four sorting techniques when executed on the server. It can be clearly seen from the graphs that execution happens at a faster rate when executed on server. It is also observed that the execution time of Quick sort and Merge sort falls almost in the same band and Insertion sort and Selection sort fall on a common band.

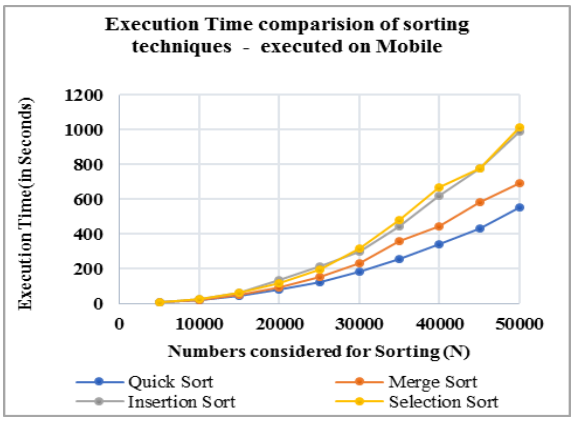

Fig.9. Execution Time comparison when executed on Mobile

Figure 11 gives the comparison of mobile energy consumption of all the four sorting techniques when executed on the mobile. Figure 12 gives the comparison of mobile energy consumption of all the four sorting techniques when executed on the server. It can be clearly seen from the graphs that all the four sorting techniques consume very less mobile energy when execution happens on server compared to execution on mobile. It is also observed that the mobile energy consumption of Quick sort and Merge sort falls almost in the same band and Insertion sort and Selection sort fall on a common band.

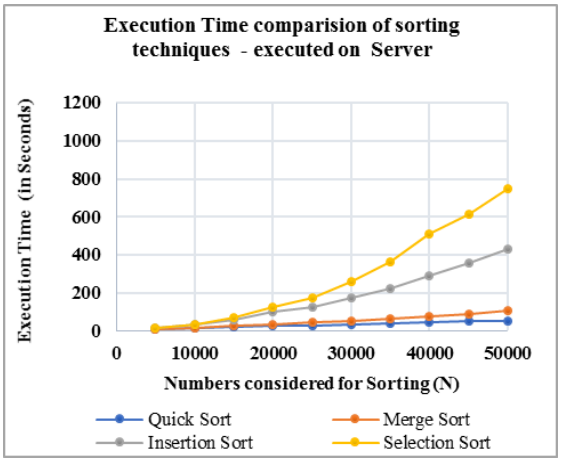

Fig.10. Execution Time comparison when executed on Server

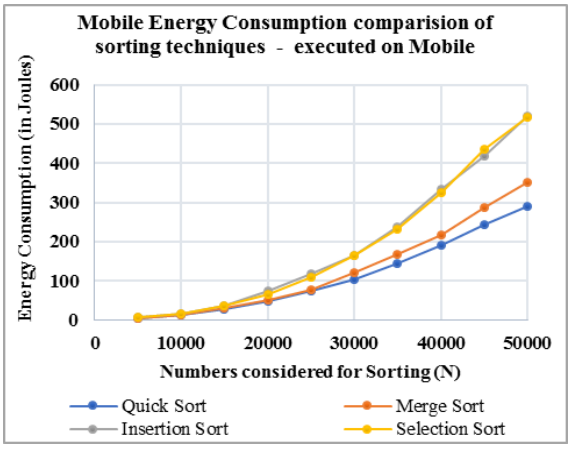

Fig.11. Energy Consumption comparison when executed on Mobile

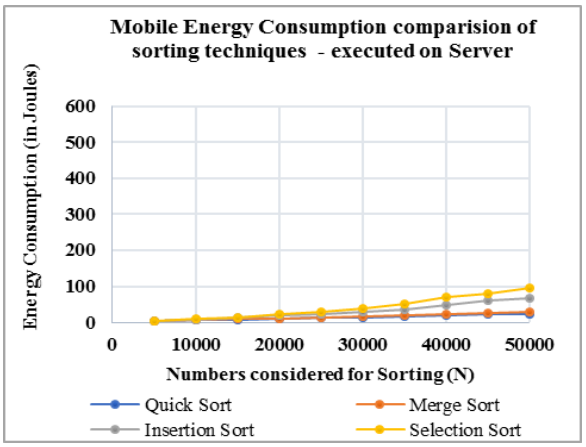

Fig.12. Energy Consumption comparison when executed on Server

It is observed that applications with less computation perform better on mobile device as it does not involve any communication overhead. However, resource or computation intensive applications would perform better

when offloaded onto the resource rich servers rather than executing on the mobile device. As the computation intensity increases, the application might crash when executed on the mobile device due to resource constraints so it is better to offload on the server.

V. Conclusion

In this paper, a comparison study was done by executing the four sorting techniques on the mobile device and the server. It was found that the performance of the mobile device increases phenomenally when all the four sorting techniques are executed on the server compared to the execution on the mobile as computation intensity increases. Also, it was observed that as input size increases, a huge difference in performance is seen when the tasks are offloaded to the server. Energy consumption of the mobile is less when the sorting techniques are executed on the server compared to the execution on the mobile. The future research work includes a decision making algorithmic approach to decide the execution of a task on the mobile or server and measure the performance based on various parameters.

References

[1] S. Abolfazli, Z. Sanaei, M. Alizadeh, A. Gani, and F. Xia, “An experimental analysis on cloud-based mobile augmentation in mobile cloud computing,” IEEE Transactions on ConsumerElectronics, vol. 60, no. 1, pp. 146–154, 2014.

[2] M Shiraz, A Gani, "A lightweight active service migration framework for computational offloading in mobile cloud computing”, The Journal of Supercomputing, 2014 - Springer

[3] Xia, Feng, Fangwei Ding, Jie Li, Xiangjie Kong, Laurence T.Yang, and Jianhua Ma. "Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing." Information Systems Frontiers 16, no. 1 (2014): 95-111.

[4] Abolfazli, Saeid, Zohreh Sanaei, Muhammad Shiraz, and Abdullah Gani. "MOMCC: market-oriented architecture for mobile cloud computing based on service oriented architecture." In Communications in China Workshops (ICCC), 2012 1st IEEE International Conference on, pp. 8-13. IEEE, 2012.

[5] Liu, Yanchen, and Myung J. Lee. "An effective dynamic programming offloading algorithm in mobile cloud computing system." 2014 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2014.

[6] Xiang, Liyao, Shiwen Ye, Yuan Feng, Baochun Li, and Bo Li. "Ready, set, go: Coalesced offloading from mobile devices to the cloud." In IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 2373-2381. IEEE, 2014.

[7] Kaya, Mahir, Altan Koçyigit, and P. Erhan Eren. "A Mobile Computing Framework Based on Adaptive Mobile Code Offloading." 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications. IEEE, 2014.

[8] Wu, Huaming, Qiushi Wang, and Katinka Wolter. "Tradeoff between performance improvement and energy saving in mobile cloud offloading systems." 2013 IEEE International Conference on Communications Workshops (ICC). IEEE, 2013.

[9] Fekete, Krisztian, Adam Pelle, and Kristof Csorba. "Energy efficient code optimization in mobile environment." 2014 IEEE 36th International Telecommunications Energy Conference (INTELEC). IEEE, 2014.

[10] Bolla, Raffaele, Rafiullah Khan, Xavier Parra, and Matteo Repetto. "Improving Smartphones Battery Life by Reducing Energy Waste of Background Applications." In 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, pp. 123-130. IEEE, 2014.

[11] Silva, Francisco Airton, Paulo Maciel, and Rubens Matos. "SmartRank: a smart scheduling tool for mobile cloud computing." The Journal of Supercomputing 71.8 (2015): 2985-3008.

[12] Elgendy, Mostafa A., Ahmed Shawish, and Mahmoud I. Moussa. "MCACC: New approach for augmenting the computing capabilities of mobile devices with Cloud Computing." Science and Information Conference (SAI), 2014. IEEE, 2014.

[13] Angin, Pelin, Bharat Bhargava, and Zhongjun Jin. "A Self-Cloning Agents Based Model for High-Performance Mobile-Cloud Computing." 2015 IEEE 8th International Conference on Cloud Computing. IEEE, 2015.

[14] Truong-Huu, Tram, Chen-Khong Tham, and Dusit Niyato. "To Offload or to Wait: An Opportunistic Offloading Algorithm for Parallel Tasks in a Mobile Cloud." Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on. IEEE, 2014.

[15] Abolfazli, Saeid, Zohreh Sanaei, Abdullah Gani, Feng Xia, and Wei-Ming Lin. "RMCC: Restful Mobile Cloud Computing Framework for Exploiting Adjacent Service-Based Mobile Cloudlets." In Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on, pp. 793-798. IEEE, 2014.

[16] http://ziyang.eecs.umich.edu/projects/powertutor/

[17] Abolfazli, Saeid, Abdullah Gani, and Min Chen. "HMCC: A Hybrid Mobile Cloud Computing Framework Exploiting Heterogeneous Resources." Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on. IEEE, 2015.

[18] Othman, Mazliza, Abdul Nasir Khan, Shahbaz Akhtar Abid, and Sajjad Ahmad Madani. "MobiByte: an application development model for mobile cloud computing." Journal of Grid Computing 13, no. 4 (2015): 605-628.

[19] Salama, Ahmed S. "A swarm intelligence based model for mobile cloud computing." International Journal of Information Technology and Computer Science (IJITCS) 7, no. 2 (2015): 28.

[20] Fernando, Niroshinie, Seng W. Loke, and Wenny Rahayu. "Honeybee: A programming framework for mobile crowd computing." International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services. Springer Berlin Heidelberg, 2012.

[21] Fernando, Niroshinie, Seng W. Loke, and Wenny Rahayu. "Mobile cloud computing: A survey." Future Generation Computer Systems 29, no. 1 (2013): 84-106.

[22] Adamuthe, Amol C., Vikram D. Salunkhe, Seema H. Patil, and Gopakumaran T. Thampi. "Cloud Computing–A market Perspective and Research Directions." International Journal of Information Technology and Computer Science (IJITCS) 7, no. 10 (2015): 42.

[23] Yang, Seungjun, Donghyun Kwon, Hayoon Yi, Yeongpil Cho, Yongin Kwon, and Yunheung Paek. "Techniques to

minimize state transfer costs for dynamic execution offloading in mobile cloud computing." IEEE Transactions on Mobile Computing 13, no. 11 (2014): 2648-2660.

[24] Chen, Chien-An, Myounggyu Won, Radu Stoleru, and Geoffrey G. Xie. "Energy-efficient fault-tolerant data storage and processing in mobile cloud."IEEE Transactions on cloud computing 3, no. 1 (2015): 28-41.

[25] Xiang, Xudong, Chuang Lin, and Xin Chen. "EcoPlan: energy-efficient downlink and uplink data transmission in mobile cloud computing." Wireless Networks 21, no. 2 (2015): 453-466.

[26] Suneel, K. S., and H. S. Guruprasad. "An Approach for Server Consolidation in a Priority Based Cloud Architecture." BVICAM's International Journal of Information Technology 8, no. 1 (2016).

[27] Sanaei, Zohreh, Saeid Abolfazli, Abdullah Gani, and Rajkumar Buyya. "Heterogeneity in mobile cloud computing: taxonomy and open challenges." IEEE Communications Surveys & Tutorials 16, no. 1 (2014): 369-392.

[28] Najmeh Moghadasi, Mostafa Ghobaei Arani, Mahboubeh Shamsi,"A Novel Approach for Reduce Energy Consumption in Mobile Cloud Computing", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.11, pp.62-73, 2015.

[29] urRehman Khan Atta, Mazliza Othman, Sajjad Ahmad Madani, and SameeUllah Khan. "A Survey of Mobile Cloud Computing Application Models.", IEEE Communications Surveys & Tutorials, Vol. 16, No. 1, 2014.

Authors' Profiles

Sindhu K is currently working as an Assistant Professor in Department of Information Science and Engineering at BMS College of Engineering, Bangalore, India. She received her M.Tech in Computer Network Engineering from Visvesvaraya Technological University. She is currently pursuing her Ph.D. at Visvesvaraya Technological University. Her research interest includes cloud computing, mobile cloud computing and mobile based application development.

H S Guruprasad holds Ph.D. in Computer Science. He is working as Professor in the Department of Information Science and Engineering at BMS College of Engineering, Bangalore, India. He has over two decades of experience in teaching field. His research interests include Networks and Communication, Cloud Computing and Internet of Things.

How to cite this paper: Sindhu K, H.S.Guruprasad, "An Experimental Analysis on Performance and Energy Saving in Mobile Cloud Computing", International Journal of Information Technology and Computer Science(IJITCS), Vol.9, No.9, pp. 45-52, 2017. DOI: 10.5815/ijitcs.2017.09.04

