MAKING THE CASE FOR COMPUTATIONAL OFFLOADING IN MOBILE DEVICE CLOUDS

PAMBALANAGESWARARAO 1, J RANJITH 2, BUCHI REDDY CHINTAKINDI 3
4.A.SANGEETHA, 5.K.MOUNIKA
Assistant Professor, Department of Computer Engineering, Ellenki college of Engineering and Technonlogy, patelguda (vi), near BHEL ameenpur (m), Sangareddy Dist. Telangana 502319..

ABSTRACT In this paper, we consider an environment in which compu- tational offloading is adopted amongst mobile devices. We call such an environment a mobile device cloud (MDC). In this work, we highlight via emulation, experimenation and real measurements, the potential gain in computation time and energy consumption that can be achieved by offload- ing tasks within an MDC. We also propose and develop an experimental platform to enable researchers create and ex- periment with novel offloading algorithms in MDCs.

Categories and Subject Descriptors
C.2.4 [Computer Systems Organization]: Computer Com- munication Networks—Distributed Systems

Keywords.Computation Offloading, Measurements, Mobile Device Clouds

1. INTRODUCTION

Computational offloading, also known as cyber foraging, had greatly evolved over the past few years [4]. Recent ex- amples of this work includes offloading tasks from mobile devices to remote cloud resources [2], automatically trans- forming mobile applications by partitioning its execution into offloadable tasks [1], and arguing the need for bring- ing computational resources closer to offloaders in order to save energy [5].

In this work, we consider environments in which computa- tional offloading is performed among a set of mobile devices forming what we call a Mobile Device Cloud (MDC). Such offloading context was also considered in Serendipity [6]. We build on their work by adopting an experimental approach highlighting the potential gain in energy and time which can be achieved by offloading computation among devices in an MDC. We investigate the possibility of saving energy and time by offloading sub-tasks to neighboring mobile devices.

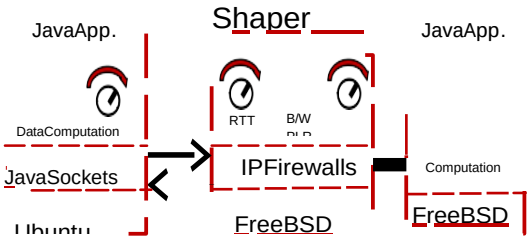

Figure 1: Emulation testbed and preliminary results for moderate intensive tasks (30MFLOP)

We implement an emulation testbed as a first step towards evaluating the potential gain of computational offloading in mobile environments. The testbed, shown in Fig. 1-(a), in- cludes an offloading application running on a client, an of- floadee server receiving tasks varying in data and computa- tional needs from the offloader, and a traffic shaper situated between the client and server emulating various communica- tion technologies. We identify five communication technolo- gies that can

be used to offload tasks, namely, Bluetooth 3.0, Bluetooth 4.0, WiFi Direct, WiFi and 3G. The testbed evaluates the gain achieved using any of the available com- municationtechnologies for different combinations of data and computation. Figs. 1-(b),(c) are a representative subset of the results that quantify the gain of offloading moder- ately intensive computational tasks to MDCs. Results for low and high computationally intensive tasks can be found in [3]. Overall, offloading to an MDC registers up to 80% savings in time as opposed to offloading to the cloud. We also observe up to 20% savings in time by offloading to an MDC as opposed to a cloudlets [5]. These results show the potential gain of computational offloading in MDCs.

The second step in our work is proposing an experimental platform, needed by the research community [6], to enable future evaluation and assessment of MDC-based solutions. In this platform, we provide an android-based API allowing future MDC experimental applications to be built. We alsocreate a testbed that measures the energy consumed by a device while performing various tasks using different com- munication technologies. In addition, we build an offloading mobile application using our API while measuring the time taken to offload tasks to other devices and receive the result. The last step in our work is utilizing the platform we built to carry out different MDC offloading experiments. These experiments conducted over several mobile devices guide us to the types of tasks that should be offloaded, and in what scenarios is it better to offload tasks to an MDC versus ex- ecuting them locally on the offloader device. Our experi- mental results show that it is possible to gain time and en- ergy savings, up to 50% and 26% respectively, by offloading within MDC, as opposed to locally executing tasks.

2. EXPERIMENTAL APPROACH

We believe there is a need for a generic flexible platform that can be utilized by researchers to freely test mobile cloud computing resource sharing and offloading solutions. This tool should decouple two main components that characterize any mobile application: the amount of data as well as the computational load that any task or job requires. These two components should also be easily broken down into distributable sub-tasks that researchers can control in real- time. Similar to simulation, this flexibility in the platform allows researchers to test their solutions over a fine-grained range of parameters that can represent a wider spectrum of current and future applications.We introduce our mobile device cloud (MDC) experimen- tal platform for mobile cloud computing research, shown in Fig. 2. We implement the MDC platform as an an- droid application, called MDCloud, based on a set of APIs for mobile-to-mobile task offloading. This platform allows users to generate tasks with different computational loads (measured in total floating point operations and denoted in MFLOP) and relevant data input (measured and denoted in MB). It also provides APIs that enable building more spe- cialized applications that can offload sub-tasks, set by the user, using various wireless technologies, such as WiFi, Blue- tooth, or WiFi Direct. The user can select the number of connected devices from a pool of devices within its proxim- ity, as well as the amount of data and computational load to be offloaded to each connected device. When the user ex- ecutes a task generation and offloading scenario by pressing the send button, the original task is, therefore, fragmented and the selected percentages will be forwarded to remote devices while the remaining sub-tasks will run locally. The MDC platform enables the application to log the total re- sponse time for each task (i.e., the time when the task was initiated to the time when the results are sent back to the initiating user). It also logs the task computational comple- tion time for

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-3, ISSUE-12, 2016
DOI: 10.21276/ijcesr.2016.3.12.6

every device as well as the data transfer time separately.

In order for the MDC experimental platform to be com- plete, real-time power measurements are needed for vari- ous states of an application. We therefore set-up an energy measurement circuit, shown in Fig. 2, in order to measure the computational and communication energy consumption. We remove the battery from the phone, solder a DC power supply to phone's power input as shown in the figure, and supply a constant voltage that matches that of the battery. We measure the electric current drawn as a result of each event, such as sending or receiving data (using Bluetooth,WiFi, or WiFi direct) as well as a given computational load, and calculate the power consumed for each event. We run these events independently for the duration of one minute to smooth out any fluctuations that occur as a result of other OS-related tasks. Once we have the power for each event computed, we finally measure the duration taken for any event in our real experiments and calculate the resulting to- tal energy consumed in Joules. We generate power usage profiles for various computational and data transmission de- mands, which are in turn used as input to the evaluation described in the next section.

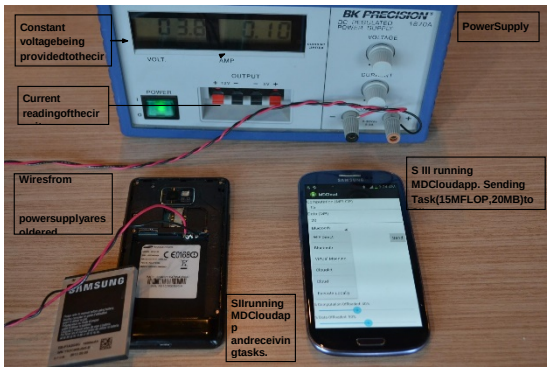

Figure 2: MDC Experimentation Platform: measur- ing energy consumption on an S2 upon receiving a task (15MFLOP, 20MB) from S3 device

	SII	SIII
WiFi-Scanning	455mW	520mW
WiFi-Idle	210mW	280mW
Bluetooth-Idle	36mW	40mW
Bluetooth-ScanningBluetooth-Sending(10MB)Bluetooth-Receiving(10MB)	540mW 370mW 296mW	400mW 280mW 240mW
WiFiDirect-Idle	none	320mW
WiFiDirect-Sending (10MB)WiFiDirect-Receiving(10MB)	none none	920mW 640mW
ExecuteTask(200MFLOP)	648mW	320mW

Table 1: Experimental energy readings on Samsung S2 and S3

Table 1 compares different energy measurements while performing wireless transfers using Bluetooth (BT) and WiFi Direct (WiFi D.) between two Samsung SII devices and two Samsung SIII devices. We send the same data sizes using both Bluetooth and WiFi Direct, and show that Bluetooth is 80% to 120% more efficient than WiFi Direct. Moreover, we notice that sending data costs 10% to 25% more energy than receiving data independent of the wireless communication used. This confirms that WiFi Direct is an energy expen- sive technology; in fact, SIII with WiFi Direct radio on and connected to another SIII, consumes almost the same energy like the SIII sending via Bluetooth to another SIII device. We also measure the computational energy consumption by running a high load task on SII and SIII devices. We show that the local execution in an SII device costs almost double the energy required to send data via Bluetooth. Inspired by these results, one may expect that offloading data to more powerful mobile devices using a low energy wireless tech- nology may be advantageous. This previous statement is highlighted by almost a 50% energy gain when using SIII to run the same task.

3. EXPERIMENTAL ANALYSIS

We create an experimental testbed consisting of two Sam- sung Galaxy SII and two Samsung Galaxy SIII phones, all running our MDCloud application. We aim to obtain prac- tical insights into making offloading decisions in mobile de- vice clouds in order to make conclusions about appropri- ate strategies to be adopted for developing

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-3, ISSUE-12, 2016
DOI: 10.21276/ijcesr.2016.3.12.6

offloading al- gorithms. In all of these scenarios, we chose Bluetooth as the standard communication technology, because of its widespread availability in almost all modern smart devices. Fig. 3 summarizes a subset of our experimental results. Additional results can be found in [3]. Ror a higher MFLOP value, we show larger gains in both energy and time con- servation while offloading the task to another device using MDCs (Figs. 3-(a),(b)). We register up to 50% gain in time and 26% gain in energy by offloading half the task to one other device. This gain is further amplified when considering higher MFLOP values [3]. We note that this result corrobo- rates the emulation experiments' results. Moreover, we show that offloading to more than one device further improves the gain with respect to time (Fig. 3-(a)). For example, we di- vide the offloaded task equally among the offloader and two offloadee devices such that each device carries out 33% of the task. We achieve up to 40% gain in time by offloading to two devices as compared to one. We can also observe that for higher data values, this gain is reduced because the over- head introduced by exchanging larger data sizes mitigates the gain achieved by distributing the computation amongmultiple devices.

We then vary the percentage of computation offloaded to the offloadee device, between 0% and 100%, in order to de- termine the optimal percentage to offload so as to maximize gains in time and energy. Determining such optimal distri- bution during runtime will be investigated in future work. Fig. 3-(c) shows the optimal time registered when 20% of the task is offloaded to the offloadee device and the rest is executed locally achieving up to 51% gain in time. In terms of energy, the best gain is registered while offloading 100% of the task to another device, yielding up to 16% conservation in energy [3].

We pose the following question: which factor (data or computation) consumes more energy during the offload op- eration? Fig. 3-(d) shows that communication can take up to 100 times more time than computation does. This is also shown when 25% increase in data costs 4x more time than a 25% increase of task complexity. This analysis shows that having more computation provides avenues for more gain to be achieved in terms of time and energy conservation, while having more data means a reduction in the energy and time that can be achieved by offloading tasks into an MDC.

4. CONCLUSION AND FUTURE WORK

In this work, we have motivated the use of mobile devices in creating mobile device clouds that can be used to save time and energy when it comes to executing computationally heavy tasks. We have shown the potential gain in both time and energy, up to 50% and 23% respectively, achieved by offloading to other devices in an MDC. These results were also corroborated by carrying out experimentation on our MDC testbed. We have presented different insights into the factors that affect the offloading decision by carrying out further experiments on our MDC testbed.

We have provided an MDC platform with an API that allows offloading algorithms to be tested on actual mobile device clouds consisting of multiple devices. We have used this platform to carry out experiments that have provided insights into how to decide whether to offload a particular task or not, and what potential strategies can be adopted for making such decisions. We have shown up to 50% gain in time and 26% gain in energy via offloading, corroborating our observations from the emulation testbed results.

While we have looked into making offloading decisions based on the contents of the task at hand, in the future, we plan to explore which device a task should be offloaded to, given

ISSN (PRINT): 2393-8374, (ONLINE): 2394-0697, VOLUME-3, ISSUE-12, 2016
DOI: 10.21276/ijcesr.2016.3.12.6

information regarding devices within an MDC. Given typical user mobility and expected inter-device intermittent connections, we plan to leverage social context and contact history between devices to determine those that are more likely to respond the fastest, and thus, result in the most efficient offloading choice. For each of the strategies used to determine which device a task should be offloaded to, we would then run simulations against actual contact history data sets to discover which strategies are most energy and time efficient in real world scenarios.

5. REFERENCES

[1] B.-G.Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti.Clonecloud: elastic execution between mobile device and cloud. In Proceedings of the sixth conference on Computer systems, EuroSys ’11, pages 301–314, New York, NY, USA, 2011. ACM.

[2] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman,

S. Saroiu, R. Chandra, and P. Bahl. Maui: making smartphones last longer with code offload. In MobiSys’10, pages 49–62, 2010.

[3] A. Fahim, A. Mtibaa, and K. Harras.Making the case for computational offoading in mobile device clouds. Technical Report CMU-CS-13-119, Carnegie Mellon University, June 2013.

[4] J. Flinn. Cyber foraging: Bridging mobile and cloud computing. Synthesis Lectures on Mobile and Pervasive Computing, 7(2):1–103, 2012.

[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies.The case for vm-based cloudlets in mobile computing. Pervasive Computing, IEEE, 8(4):14–23, 2009.

[6] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura. Serendipity: enabling remote computing among intermittently connected mobile devices. In MobiHoc, pages 145–154, 2012.