Reaching for the clouds: contextually enhancing smartphones for energy efficiency

Radu-Corneliu Marin
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
Bucharest, Romania
radu.marin@cti.pub.ro

Ciprian Dobre
Faculty of Automatic Control and Computers
University Politehnica of Bucharest
Bucharest, Romania
ciprian.dobre@cs.pub.ro

ABSTRACT

Energy efficiency has gradually become a compulsory need in mobile computing as the processing requirements for smartphones have increased exponentially. Moreover, the current demand is stretching beyond the extents of modern battery technology. In this sense, we introduce a novel collaboration solution for mobile devices based on a contextual search entitled Hybrid Contextual Cloud for Ubiquitous Platforms comprising of Smartphones (HYCCUPS). HYCCUPS takes advantage of the pervasive nature of smartphones and of current wireless communication technologies as to offer offloading execution of mobile applications in an opportunistic on-the-fly hybrid computing cloud. We design an adaptive contextual search algorithm for schedulling mobile code execution in smartphone communities based on predicting the availability and mobility of devices in the near vicinity. We emulate the HYCCUPS framework based on real user traces and we prove that it improves battery health, maximizes power save, minimizes overall execution time of mobile applications and it preserves or even enhances user experience.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*distribution functions, statistical computing, time series analysis*

Keywords

ubiquitous computing, predictability, contextual search, mobile computing, energy awareness

1. INTRODUCTION

The advent of smartphones has paved the way for developers and researchers alike to create rich context-aware systems, thus permanently remodelling the ubiquitous computing community. The main popularity factor of smartphones has always been their cutting edge hardware (fast CPUs, extensive memory and storage) coupled with the access to high speed mobile networks (WiFi, 3G/4G). Thus, such mobile devices pose a great opportunity for human dynamics researchers, as human behavioural patterns have already been extracted and analyzed in previous studies using real mobile user traces [4].

The continuous development and improvement of smartphone platforms have yielded a most important issue which seems to have eluded mobile Original Equipment Manufacturers (OEMs) and software engineers alike, namely *extreme energy consumption*. Such a concern could have been ignored or even gone unnoticed if it hadn't overlapped with another pressing matter: battery manufacturers are struggling with the physical limit of current technologies. These two concurrent factors are endangering the reputation of both mobile hardware and software providers, as customer dissatisfaction is increasing. It seems that the excelling computing power and the myriad feature sets of smartphones are being outshined by low battery life. Most current directions taken by the mobile computing community in regard to energy efficiency are focused at throttling resources by means of complicated techniques that focus on individual hardware components: the CPU [8], the display [12] or network transfers [1]. Although such techniques provide undeniable power reduction, their applicability will prove to be limited, as smartphone requirements start to grow even larger, and they seem to be increasing at *exponential* rates.

This paper describes a novel solution towards smartphone energy efficiency, namely HYCCUPS. Our solution provides the means for mobile devices to interact and collaborate over wireless networks as to create a novel hybrid computing cloud in which mobile terminals are clients, as well as computing resources. By doing so, the hybrid cloud acts as a middleware that enables smartphones to schedule and offload execution onto other resources as to reduce energy consumption, and save battery life. Scheduling is aided by a contextual search method which predicts the availability and mobility of mobile resources in order to optimize remote workloads and reduce the number of speculative remote tasks.

Preliminary versions of our work were previously published in [10] and [9]. This paper adds more extensive work by introducing and thoroughly explaining the contextual search algorithm used in the HYCCUPS framework. Furthermore, we emulate HYCCUPS based on real user traces and we analyze the performance of the entire framework.

Moreover, we use the methodology and guidelines from [4] to prove the correctness and feasability of HYCCUPS.

The remainder of this paper is organized as follows. Section 2 presents the state-of-the-art for solutions similar to HYCCUPS. Section 3 describes the inner-workings of HYCCUPS as a conceptual framework. Section 4 analyzes the correctness and performance of our solution by emulating the framework based on real user traces. Section 5 concludes the paper.

2. RELATED WORK

In the past few years there has been a growing interest in the field of human dynamics starting with the work of Barabasi [2]. Since then, many research areas benefited from such endeavours, one of them being mobile cloud computing (MCC). The main idea behind mobile cloud computing is to offload the execution of power-hungry tasks, and move the data pertaining to said computation from mobile devices onto clouds with respect to the intrinsic mobility of mobile users and human behavioral patterns. Dinh et al. [6] point out the main advantages of using MCC: extending the battery lifetime by moving computation away from mobile devices, better usage of data storage capacity and of available processing power, improving the reliability and availability of mobile applications. Furthermore, MCC inherits the benefits of cloud computing as well: dynamic provisioning and scalability.

The benefits of using such solutions must be weighed with the security threats of cloud computing. The main concern of MCC is the lack of privacy in storing data on clouds. Bisong et al. [3] advise developers in MCC not to store any data on public clouds, but to turn their attention towards internal or hybrid clouds. In this sense, Marinelli [11] proposes the use of smartphones as the resources in the cloud and not as mere clients of it. He introduces Hyrax, a platform derived from Hadoop which offers cloud computing for Android devices, and allows client applications to access data and offload execution on heterogeneous networks composed of smartphones. Interestingly enough, Hyrax offers these features in a distributed and transparent manner with respect to the developer. Although Hyrax is not primarily oriented at reducing energy consumption, but at fully tapping into distributed multimedia and sensor data without large network transfers, it represents a valuable endeavour due to its general architecture in which smartphones are the main resources offered by the cloud. Moreover, Huerta-Canepa et al. [7] offer a similar solution of offloading computation on an on-the-fly ad hoc virtual cloud computing provider comprised of mobile devices. Also based on Hadoop, it extends the premises of Hyrax by deciding whether to execute code remotely or locally based on the vicinity and detected activity of nearby nodes.

Unlike previous solutions, HYCCUPS proposes the creation of a hybrid computing cloud in which the main resources are smartphones and focuses on reducing the overall consumed power of the interconnected devices. The main differentiating factor of HYCCUPS from other platforms is that the decision of whether to run a task remotely or locally is taken by a contextual search method based on a real life trace model. Moreover, the context takes into consideration both mobility and availability of a node to offload a task and it also does not neglect the overall quality of experience of the users which are part of the cloud.

3. HYCCUPS

HYCCUPS is a ubiquitous computing conceptual framework which proposes to offer smartphone devices the opportunity to collaborate over high-speed wireless networks in a distributed and transparent manner in order to reduce the overall power consumption. The novelty of the solution stems from the use of smartphones as both the computing resources, but also as the clients in a hybrid cloud. As such, the framework was designed as to fully cover the four issues stated by Marinelli [11] in taking such an endeavour: (1) each node is owned by a different user; (2) each node is likely to be mobile; (3) the network topology is highly dynamic; and (4) each mobile node is battery powered. These issues influenced HYCCUPS' design. Furthermore, HYCCUPS attempts to solve the issues pointed out by Cuervo et al. [5] in the MAUI system: (1) We use WiFi instead of mobile networks (3G) because it is more energy-efficient; (2) cloud resources are closer to the mobile devices as to reduce the network transfers.

Considering that a mobile cloud is, by definition, always in perpetual motion, and its components are usually owned by multiple users, the availability of a node to be able to process a submitted workload is uncertain. As such, HYCCUPS uses a contextual search method to determine if a workload should run remotely or locally, and acts as a medium of offloading the current task to the best candidate in the mobile cloud as to reduce the power consumption while preserving the quality of experience. Also, considering the heterogeneity of a cloud, a node needs to be able to update its trained data by means of a feedback process responsible for evolving the system and guaranteeing better availability resolution.

3.1 The architecture

The HYCCUPS framework provides a middleware layer on top of which developers seamlessly design, build and deploy applications which will take advantage of the full potential of contextual execution offloading. The overall architecture of HYCCUPS is presented in Figure 1.

The right hand side of Figure 1 represents the hybrid cloud deployment over a WiFi network. There are three types of cloud terminals: The **Mobile Terminal (MT)** is any smartphone terminal not plugged into an electrical outlet. It's main characteristic is mobility and it isn't always available for executing workloads. A **Fixed Mobile Terminal (FMT)** is any plugged-in smartphone terminal. It acts as a Fixed Terminal, but may rapidly return to a mobility state (MT). Finally, the **Fixed Terminal (FT)** is a wearable computer that is available for executing workloads. This is an optional component, as it raises compatibility issues: the developer has to explicitly redesign the mobile application code so it can be deployed on other platforms.

The left hand side of Figure 1 illustrates the anatomy of a MT (because FT and FMT do not impose any issues, they act as regular cloud computing resources). The main components of a MT are the following: The **HYCCUPS Tracer** is an application designed to collect contextual data from smartphones. The **trained availability data records** are kept in persistent datastores that aggregate the contextual information gathered by the HYCCUPS Tracer. The **WiFi daemon** is in charge of creating and maintaining the WiFi bus, and it is also responsible for resource discovery. The component is the actual glue holding the entire cloud together. The **Availability checker** is

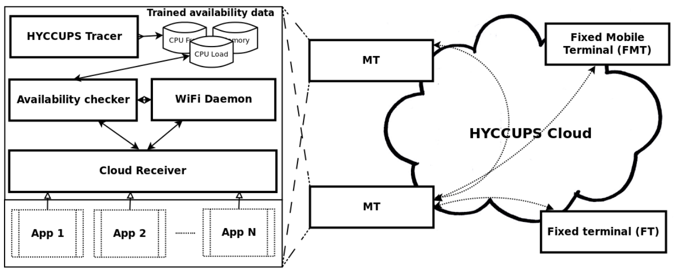


Figure 1: HYCCUPS architecture.

responsible for conducting the contextual search that determines if a task should be offloaded and/or if an incoming remoteable task should be executed locally. The **Cloud receiver** is responsible for resolving the execution runtime of incoming/outgoing remote tasks. If the decision of offloading is done by the Availability checker, the actual enforcing of that decision is done by the Cloud receiver.

We take the following features into consideration as tracing data: *mobility* (we are more interested in wireless mobility than geographical tracking; as such, we aim at tracing patterns of users interacting over WiFi), *availability* (we collect statistical information about available resources such as CPU frequency, CPU load, available memory and battery statistics), and *quality of experience* (we are interested in discovering when the user is performing operations on the device, e.g. opening/closing applications).

The machine learning algorithm in the Availability Checker is responsible for aggregating tracing data into a predictor which is able, based on the timing interval, to determine that the device is available to offload a task, that the device is stable mobility-wise for a sufficient interval as to process the remote task, and that the owner of the remote device is not experiencing difficulties due to the executing task. We previously showed [10] the capability of the predictor to extract synergic patterns, and we explored the limits in foreseeing the whereabouts of individuals in relations with the virtual communities they are part of. Our initial premises were that synergic patterns in academic and office environments are subject to repeatability. In such environments, we proved that human behaviour is predictable, as this is the foundation of our framework.

The basic behaviour of a MT is to attempt to offload execution onto the cloud and respond to other workloads submitted into the cloud. There are two typical scenarios:

1. an application submits a workload onto the cloud. Based on the Availability Checkers' offloading decision, the Cloud Receiver passes the task either to the WiFi daemon or to the local application. Because of the middleware layer, the user will not know if the submitted workload was offloaded, or if it actually executed locally. The mobile cloud does not impose that all execution be done on other devices, it only attempts to schedule it to run on the device that will maximize both energy efficiency and user experience.

2. the Cloud Receiver receives a task from the cloud, it assesses that it can execute the workload (via the Availability Checker) and accepts or declines accordingly. If more than one device accepts a workload, the availability will be the tie-breaker (FT and FMT will always have highest availability).

3.2 The contextual search algorithm

Contextual search is a technique used in pervasive systems to augment a query with context data from users in order to obtain content optimized for a specific situation [13]. In order to perform contextual search, multiple phases are required:

- sensing contextual data: gathering or tracing raw context data from sensors in the environment;
- aggregating and recording traced data: aggregating raw context data into higher level context and persistently storing it;
- conducting the context-enhanced search: augmenting queries by means of the contextual records.

The context retrieval is provided by the HYCCUPS Tracer. The following features are considered: CPU load, CPU frequency, available memory, interactions with peers over WiFi, battery level, battery charging state, and user activity events. The tracing data is aggregated into trained availability records. Each record behaves similar to a circular buffer as it contains aggregated contextual data for each day of the week with a given sampling rate. We consider a week to be sufficient, as human behavioral patterns in academic and office enviroments tend to occur within this interval. When storing a feature value, it is aggregated with the previous tracing data at the closest sampling point smaller than the timestamp of the event. Based on the aggregation method, there are two types of records:

1. **Averaging records**: the data stored in these records is a continuous real value and it is aggregated by using the average;
2. **Probability records**: these records contain boolean events and the aggregation method is the probability that the event is true.

The sampling intervals were chosen by inspecting the inter-event times for each feature, and also by taking into consideration the mobile development aspect. Given that HYCCUPS will be deployed onto smartphone platforms, the size of the records is important. Therefore, in the cases of CPU load and frequency, we made a compromise between informedness and spatial restraints by choosing a 1 minute sampling interval. Although the CPU features are less reliable, they still offer valuable quantitative information.

As long as HYCCUPS is running, the records are continuously updated, so the contextual search algorithm adapts to environmental changes (e.g.: user changes her patterns or battery health degrades). However, these averaged momentary values have no meaning by their own. To make use of them, we extract temporal patterns over intervals of execution, so that the system can predict future values for them.

To predict the values for features on any given interval of time, HYCCUPS uses univariate linear regression to determine a hypothesis h of the form $h(x) = \theta_1 + \theta_2 x$ which is able to approximate values for features for any given timeframe. The predictor for the average of a feature over the interval $[currentTime, currentTime + \Delta t]$ is of the following form: $predictFeature(\Delta t)$ and it computes this future value as follows:

1. construct the statistical analyzed interval: $[currentTime - \Delta t, currentTime + 2 * \Delta t]$; we triple the predicted interval as to provide sufficient information for the actual prediction.
2. extract the n samples from the feature's record in the statistical analyzed interval. The records are considered to be circular (the weeks wrap around).
3. construct the X matrix, where x_i is the i-th sampling point in the analyzed interval.

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ ... & ... \\ 1 & x_n \end{pmatrix}$$

4. construct the y column vector, where y_i is the i-th feature value.
5. compute the Θ column vector, by solving the following equation:

$$X * \Theta = y \tag{1}$$

   The Θ column vector contains the parameters of our heuristic h which is able to predict values of a feature. In order to find Θ, HYCCUPS uses the normal function:

$$\Theta = (X^T \times X)^{-1} \times X^T \times y \tag{2}$$

6. compute the average of the feature in the prediction interval: $h((currentTime + \Delta t)/2)$.

Given that such a system is constantly adapting to environmental changes, it is only natural to assume that predictors may output erroneous values. In this sense, the prediction output is weighed against the degree of informedness and the predictor accuracy:

$$\begin{aligned} predictFeature(\Delta t) = w \times h(currentTime + \frac{\Delta t}{2}) + \\ (1 - w) \times momentary\_value \end{aligned} \tag{3}$$

where w is the feedback weight:

$$w = \frac{informedness}{2} + \frac{accuracy}{2} \tag{4}$$

The degree of informedness expresses how many samples have been recorded in the statistical analysis interval:

$$informedness = \frac{no\_filled\_samples}{no\_total\_samples} \tag{5}$$

In order to measure the efectiveness of the predictor, HYCCUPS uses the balanced accuracy:

$$\begin{aligned} accuracy = \frac{1}{2} \times \frac{positives}{positives + false\_positives} + \\ \frac{1}{2} \times \frac{negatives}{negatives + false\_negatives} \end{aligned} \tag{6}$$

3.3 Taking decisions based on contextual search

The Availability Checker is at the heart of the HYCCUPS framework as it implements the contextual search algorithm based on which the decision to offload a task or run it remotely is taken. Its main responsibilities include *workload profiling* (i.e., ascertaining the amount of resources needed to actually run a task), and *offloading decision* (i.e., determining if a task is better run locally or on a different device).

Given that any task can run on multiple devices, each of them having specific traits, workload profiling needs to first determine which resources are required by the task (and to what extent), compute a workload score (or computing potential) which uniquely describes the computational requirements of the task, and normalize the workload score to compare executional needs of a task for multifarious smartphone platforms. For this we introduce the *computational potential* (or simply, potential) which we define as follows: *the computing potential is the time required by the processor to run a workload considering current and predicted values for CPU load and CPU frequency in the nearby future.*

In essence, the potential is a measure of the time a task requires CPU time on a specific device. At a first glance, this measure may seem inequitable, as devices with better performing CPUs might seem disadvantaged. But, as presented further on, time turns out to be a good invariant in our problem. Let us consider an example: two devices - a low end device (A) and a high end device (B) with a 5 times faster CPU are part of the hybrid cloud; device (A) shortly becomes overwhelmed by the tasks it's executing so it offloads 5 tasks with $potential_A = 1s$ to (B) which evaluates them at $potential_B = 0.2s$ so, instead of, (A) executing them in 5s, (B) runs all of them in 1s; not much later, the user of device (B) becomes active and launches a task which (B) evaluates to $potential_B = 1$ and offloads it to (A) which evaluates it to $potential_A = 5$. Now both devices have cleared their debt and were able to run more efficiently (while the remote device was executing the offloaded task, the local device was able to execute other tasks). One should keep in mind that the workload of a task is usually directly proportional to the computational possibilites of the device that it's running on (because users tend to use applications that are suited for their device). As such, tasks from low end devices such as (A) are usually much smaller than tasks running on high end devices (B).

Furthermore, in order to reduce the number of leechers in the system, a cost model was put in place. Each time a node offloads a task for another, it remembers the debt it is owed by that node as the sum of negative potentials; also the node which is requesting the offload stores the debt it ows the other node as the sum of positive potentials. When a node responds to an offload request, it will advertise that it's potential is: $potential = true\_potential - debt\_for\_requestor$. This ensures altruism as it discourages nodes to take advantage of other nodes, and it encourages nodes with debt to offload in order to reduce their debt.

To compute the potential of a task on a device, the following steps are required to determine the potential given the number of CPU cycles estimated for the task:

1. determine the maximum acceptable time (MAT) to run a task (the time needed to run a task at lowest frequency with 50% CPU load):

$$MAT = num\_cycles \times \frac{1}{0.5 \times min\_frequency} \quad (7)$$

2. determine the potential of a task (the predicted time to run a task in curent conditions):

$$potential = \frac{num\_cycles}{predictCPULoad(MAT) \times predictCPUFreq(MAT)} \quad (8)$$

Although it's not the absolute maximum time of running a task on a device, the maximum acceptable time represents the absolute maximum time that we would allow an offloaded task to run on a device. It's more of a quantitative approach to determine the interval of time that a task should be executed in. As such, we compute the actual potential by predicting the CPU performance over the MAT interval. Therefore, the maximum acceptable time is computed only to determine the interval of prediction for the potential.

The actual importance of contextual search in HYCCUPS is: when an application generates a task to be executed in the hybrid cloud, it is dispatched to the Cloud Receiver, which forwards it further to the Availability Checker and waits for it's decision. There are actually two types of decisions that need to be taken:

- **Local decision**: the local Availability Checker decides wether it is best to execute locally (it is charging) or wether it should be offloaded onto another device;
- **Remote decision**: a remote node receives the offloaded task request and it's Availability Checker needs to determine if it can handle that remote task.

1. Local decision: Before offloading a task into the cloud, a node first decides through contextual search if the task actually needs to be executed remotely, or it is better run locally. Figure 2 depicts the decision tree for local resolution of tasks. Each decision node in the tree is actually a predictor (which uses the algorithm previously presented). If the Availability Checker decides to remotely execute the task, it sends a task offload request to all connected peers, and waits for their potentials to run the task. After gathering all of the results, it adds the local potential to the list as well and chooses the best candidate which will use minimal potential to execute the task. The task can still be run locally, if the local node has the smallest potential. It should be noted that when the battery is charging the Mobile Terminal acts as a Fixed Terminal and directly executes the task locally.

2. Remote decision: When a node receives an offload request, the Availability checker decides whether it will accept to run the task, or if it will reject it because it is unable to spare the resources. The contextual search in the remote decision is illustrated in Figure 3. Similar to the local decision, each node in the decision tree is a predictor. The remote decision is more intricate than the local one, as each node wants to maximize the number of offloaded tasks, while minimizing the offloading it's actually doing. However, the cost model previously presented is refereeing this process, so that no node will be taken advantage by others. Also similar

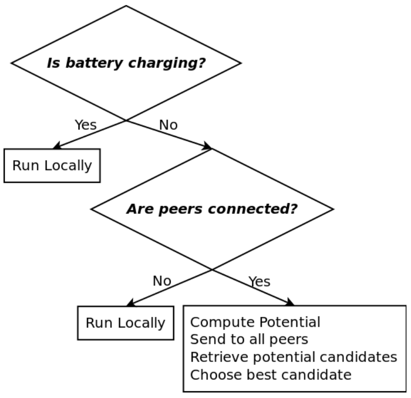


Figure 2: Local decision for offloading.

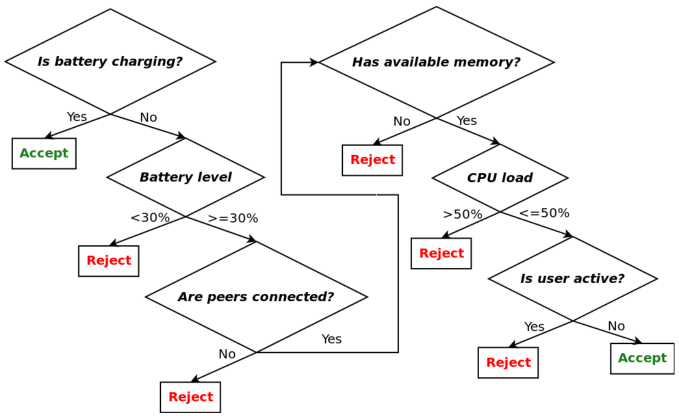


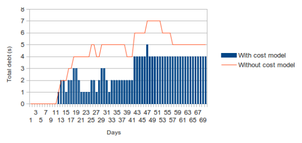
Figure 3: Remote decision for offloading.

to the local decision, when a node is charging it will attempt to act as a Fixed Terminal, rather than a Mobile Terminal, and it will advertise only a tenth of it's potential to execute the task. The main reason behind this is to encourage other nodes to offload to it while it is still charging.

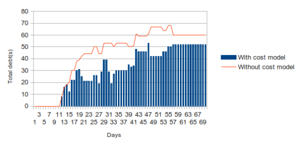
4. EXPERIMENTAL RESULTS

We have developed the HYCCUPS Emulator, which provides valuable insight into the innerworkings and performances of the entire framework. It makes use of traced contextual data such as the ones publically available in UPB2012, Rice and others (for a complete analysis, please see [10]). The HYCCUPS Emulator is implemented in Java and it uses the Efficient Java Matrix Library (EJML) in order to compute linear regression by using the normal function. Also, the emulator was built with the following assumptions:

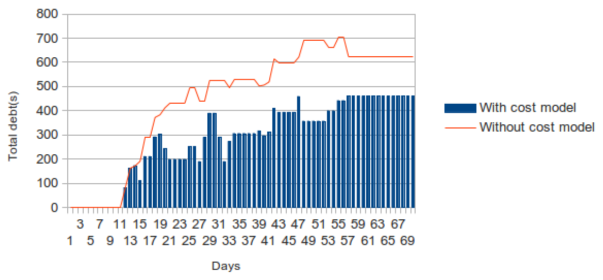
- all devices use the same reference for time: due to the fact that the devices involved in the tracing experiment did not use a network clock we must assume that they all are synchronized to the same time reference.
- all environmental events are synchronous and sequentially executed for all users. As such each event for any device acts as a barrier for all users. By so doing we are able to implement deterministic heuristics for generating repeatable workloads.



(a) 100 Mcycle tasks.

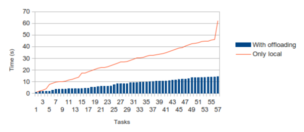


(b) 1 Gcycle tasks.

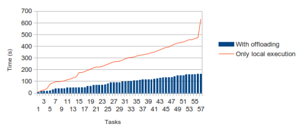


(c) 10 Gcycle tasks.

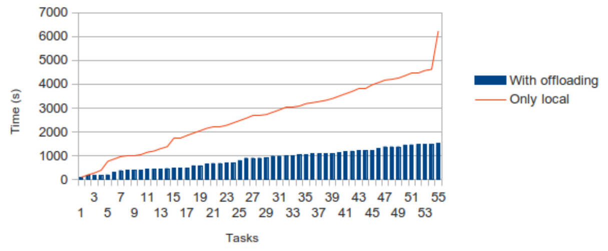
Figure 4: Total accumulated debt.



(a) 100 Mcycle tasks.



(b) 1 Gcycle tasks.



(c) 10 Gcycle tasks.

Figure 5: Total accumulated execution time for offloaded tasks.

- there is no network transfer delay: the emulator does not assume that workloads are sending data as well, instead it presumes that the data is already present on the device. Therefore all workloads are expressed in CPU cycles.
- all devices have only a single core: the tracing data did not include anything about the number of available cores or of the numbers of online cores.
- AllJoyn is functioning correctly: all interactions between peers are symetrical.
- the bandwidth of the network is considered to be unlimited.

In order for the mobile cloud to react seamlessly and coherently, an efficient Android peer-to-peer communication solution was also needed as to fulfil all responsibilities of communication in the cloud. In this respect, we selected AllJoyn as our communication framework. AllJoyn is an open source software framework which offers a peer-to-peer communication environment for heterogeneous distributed systems across different device classes over WiFi access points, with an emphasis on mobility, security and dynamism by implementing the D-Bus protocol. It provides an abstraction layer allowing it to run on multiple operating systems which enables us to register wearable computers into the mobile cloud as Fixed Terminals. We previously showed (see [4]) that, despite its many benefits, Bluetooth proves to have considerable drawbacks relating to our needs.

In the emulation process, workloads are generated in a deterministic manner so as to be repetitive using a heuristic that takes into account the following features: user activity and AllJoyn connectivity.

The main difficulty with working with the tracing data set from the HYCCUPS experiment was the lack of conscienciousness of the volunteers. Because of this, there are not many nodes actively involved in offloading, as only three devices out of all 65 participants seem to be collaborating in an energetic manner.

We have attempted to cover multiple types of workloads while analyzing the HYCCUPS framework. We have generated three scenarios using a total of 726 tasks over 2 months of emulator execution which are sent in a pseudo-random fashion to all active users in the system, with the following computational requirements:

1. 100Mcycle tasks: small tasks which are easily executed by any node. Such tasks can be associated to computing a move in a game of chess.
2. 1Gcycle tasks: regular sized tasks which should be executed by most devices. This scenario could be associated with computing the value of π.
3. 10Gcycle tasks: large tasks which should be a burden for all devices. These tasks could relate to the analysis of satellite images (given that the images are already downloaded).

Figure 4 presents the total debt accumulated by all users on a daily basis for all three scenarios. As can be seen in all three cases, the cost model gives better performance, as it seems that altruism serves the community better than leeching off of other nodes. Furthermore, Table 1 shows that even though working without a cost model can actually offload more, the overall benefit of using such a model is much greater as it actually saves more computing time.

As seen in Table 1, HYCCUPS manages to save valuable computing which is actually proportional to size of the task. Although at a first glance the amount of saved time does not seem to be large, only three devices were actively offloading tasks. This should generally change the perspective over the results. Also, adding that more than a quarter of the tasks were executed remotely on devices that are charging, the actual computing power save increases. Moreover, out of the 57(55) offloaded tasks, none of them failed to complete, which proves that the adaptive and predictive models used in collaboration with the contextual search algorithm are working properly and correctly.

Last, but not least, Figure 5 illustrates the total execution time as tasks are sequentially run. Moreover, they compare the HYCCUPS execution model with the current traditional Android model. As can be observed, the HYCCUPS framework speeds up mobile applications by means of offloading

Scenario	Model	Total offloaded	Failed	FMT offloads	Total saved time (s)
100 Mcycle	w/ cost model	57	0	16	48
	w/o cost model	57	0	16	45
1 Gcycle	w/ cost model	57	0	15	472
	w/o cost model	57	0	16	448
10 Gcycle	w/ cost model	55	0	15	4723
	w/o cost model	57	0	16	4481

Table 1: Offloading statistics for all three scenarios

onto remote idle nodes. However, these figures do not contain the execution for all of the 726 tasks, but only for the 57(55) offloaded tasks. We are not interested in the tasks that are always run locally as HYCCUPS can only optimize workloads that actually have the possibility to be offloaded (although the predictors might have missed some offloading opportunities).

5. CONCLUSIONS

We believe that current models in the design, development and deployment of mobile applications are becoming inadequate as they do not take sustainability into account. Furthermore, we consider the endeavours currently taken in mobile cloud computing to be incomplete and limited. As such, the need for novel computational models that focus on power saving is growing exponentially.

HYCCUPS introduces intelligent collaboration, a new computational model which offers feasible synergy between interacting peers by correlating mobility with availability information by means of a contextul search algorithm. Moreover, the algorithm is adaptive to environmental changes so as to improve battery health, maximize power save, minimize overall execution time of mobile applications and, last, but not least, to preserve or even enhance user experience.

We proved the correctness, feasability and performance of our solution by emulating the solution based on the aforementioned traced contextual data, and we obtain that not only does the overall execution time of tasks decrease, but also the utilization of resources in our ad-hoc cloud is optimized, thus reducing the power consumption of active peers.

6. ACKNOWLEDGMENTS

This work was supported by project "ERRIC - Empowering Romanian Research on Intelligent Information Technologies /FP7-REGPOT-2010-1", ID: 264207.

7. REFERENCES

[1] B. Aggarwal, P. Chitnis, A. Dey, K. Jain, V. Navda, V. N. Padmanabhan, R. Ramjee, A. Schulman, and N. Spring. Stratus: energy-efficient mobile communication using cloud support. *ACM SIGCOMM Comp. Communication Review*, 41(4):477–478, 2011.

[2] A.-L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207–211, 2005.

[3] A. Bisong, M. Rahman, et al. An overview of the security concerns in enterprise cloud computing. *arXiv preprint arXiv:1101.5613*, 2011.

[4] R. I. Ciobanu and C. Dobre. Predicting encounters in opportunistic networks. In *Proceedings of the 1st ACM workshop on High performance mobile opportunistic systems*, pages 9–14. ACM, 2012.

[5] 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*, pages 49–62. ACM, 2010.

[6] 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*, 2011.

[7] G. Huerta-Canepa and D. Lee. A virtual cloud computing provider for mobile devices. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, page 6. ACM, 2010.

[8] Y. Liang, P. Lai, and C. Chiou. An energy conservation dvfs algorithm for the android operating system. *Journal of Convergence*, 1(1):93–100, 2010.

[9] R.-C. Marin. Hybrid contextual cloud in ubiquitous platforms comprising of smartphones. *International Journal of Intelligent Systems Technologies and Applications*, 12(1):4–17, 2013.

[10] R.-C. Marin, C. Dobre, and F. Xhafa. A methodology for assessing the predictable behaviour of mobile users in wireless networks. *Concurrency and Computation: Practice and Experience*, 2013.

[11] E. E. Marinelli. Hyrax: Cloud Computing on Mobile Devices using MapReduce (Masters Thesis), 2009. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

[12] T. K. Wee and R. K. Balan. Adaptive display power management for oled displays. In *Proceedings of the first ACM international workshop on Mobile gaming*, pages 25–30. ACM, 2012.

[13] J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: A review. *Perv. and Mobile Computing*, 8(1):36–66, 2012.