The Mobile Cloud – more than a cloud

Michael J. O'Sullivan, Dan Grigoras
Department of Computer Science,
University College Cork, Cork, Ireland
{m.osullivan, grigoras}@cs.ucc.ie

Abstract: Nearly one billion **s**mart mobile devices are now used for a growing number of tasks, such as browsing the web and accessing online services. In many communities, such devices are becoming the platform of choice for tasks traditionally carried out on a personal computer. However, despite the advances, these devices are still lacking in resources compared to their traditional desktop counterparts. Mobile cloud computing is seen as a new paradigm that can address the resource shortcomings in these devices with the plentiful computing resources of the cloud. This can enable the mobile device to be used for a large range of new applications hosted in the cloud that are too resource demanding to run locally. Bringing these two technologies together presents various difficulties. In this paper, we examine the advantages of the mobile cloud and the new approaches to applications it enables. We present our own solution to create a positive user experience for such applications and describe how it enables these applications.

1. Introduction

Due to their large adoption rate, mobile devices such as smartphones and tablets are now increasingly the first-choice computing platform of users to access online services such as social networks and e-Commerce, and carry out routine tasks such as browsing the web, checking email, and consuming media. Mobile applications can be native or downloaded by the user from an application store such as Apple AppStore. Alternatively, HTML5 is emerging as a new version of HTML with features that allow webapps to be optimised for mobile devices.

A new approach to consumer utilisation of applications and services is mobile cloud computing, the access of cloud services from a mobile device. One major benefit of this model is that resource scarce mobile devices can avail of the in-excess computing and storage resources of the clouds. Thus, for example, mobile devices can transfer some of their work to the mobile cloud. The second benefit is that mobile clients don't need to download apps anymore if they can run similar applications and services hosted by the clouds. Finally, there is no need of synchronisation between the mobile device and the cloud during the execution of the service or application it invoked. CISCO Internet Business Solutions Group believes that the mobile cloud service market will reach $60 billion in 2016, up from $43.5 billion this year [1]. Mark Beccue, senior analyst with ABI Research, says "by 2014, mobile cloud computing will become the leading mobile application development and deployment strategy, displacing today's native and downloadable mobile applications" [2].

However, we consider that the mobile cloud model will be largely adopted by the general public and therefore successful only if mobile users will have a positive experience in all respects. The user experience involves several key aspects such as the latency between the mobile client and the cloud, the bandwidth utilisation, disconnected operation, seamless mobility management, presence of an adaptive thin client for the mobile user (UI), privacy and cost. Therefore, a positive user experience can be provided by a mobile cloud system that has cost-effective solutions for the above aspects. While there are many projects tackling different aspects of the user experience of the cloud, there are none considering it as the main objective. In an era where people in all walks of life from business to education expect to remain connected to computing resources at all times while out and about through their mobile device, the approach of our project will be crucial. More users are going to turn to

their mobile device first rather than their desktop computer, and they will want the same quality of service and reliability they get from desktops. Disruption from services, as well as high time, energy and monetary cost to access the services, will not be acceptable.

The aim of this paper is to first discuss existing approaches in Section 2, then introduce our approach in Section 3, discuss its main benefits in Section 4, and then elaborate, in Section 5, on the mobile cloud application potential.

2. Similar Work

Probably, the most well-known mobile cloud approach is the cloudlet [3] that aims at reducing the latency and bandwidth for mobile cloud applications. Cloudlets are deployed in areas like a coffee shop or an airport and consist of computers available behind the access point. The user is then only one Wi-Fi hop away from the Cloudlet, where some of the tasks could be transferred and executed. However, the cloudlet overhead is large and can be considered only as a surrogate.

Another approach is to split the application graph into components that can then be distributed in the cloud, other devices in the proximity and the mobile device itself. Such a system is Alfredo [4]. This model requires user knowledge of the application and skills to distribute its components. Obviously, this will not help the general public experience of the mobile cloud.

One suggestion is to use the mobile devices as thin clients that display results produced by the cloud [5]. However, remote viewing is not considered optimal for mobile devices, considering aspects as delays, caching, and the touch-screen interface.

It is possible to consider offloading to the cloud of demanding applications [6-9]. It requires a permanent connection to the cloud infrastructure. Typically, an optimization solver makes the decision for code to be either executed locally on the mobile device, or remotely on the cloud infrastructure. Profiling has an adverse effect on the user experience, requiring time and energy. Code offload also depends on the quality of the network. If the network connection drops while code has been offloaded, then the code that was being executed remotely must now be re-executed on the device, wasting time and resources.

3. The Cloud Personal Assistant Middleware

3.1 Objectives

The mobile cloud paradigm, while very promising, is difficult to implement in practice and faces several challenges. Maintaining a continuous connection between the mobile and the cloud, as suggested by many projects, is difficult, as the mobile device will often be in a state of change between networks, and at times will be completely disconnected. The sending and receiving of data between the mobile and the cloud is demanding on the limited battery power, even more so over a wide area 3G network, where latency is high and bandwidth is low. High latency will render some real-time applications un-usable. All of these outstanding issues result in a negative user experience of mobile cloud applications. Our mobile cloud middleware work seeks to reduce or even eliminate the impact of these problems.

3.2 The Cloud Personal Assistant

Our approach is a cloud-based middleware system that is deployed in the cloud, sitting between the mobile user and the cloud. The main concept of this system is a new cloud service, called the Cloud Personal Assistant (CPA) [10]. Similar to a business person, who delegates many tasks to the personal assistant, computing tasks can be transferred for execution to the CPA. However, the digital PA, running in the cloud, can be even more

effective than a person as it is available 24/7. This is more than just a cloud where a service is requested directly by the user in a client-server fashion. The cloud, through the CPA, is becoming more active by operating services quasi-independently from the user - see Figure 1.

The CPA instance is created and allocated to the user when s/he subscribes to the system. From that moment, the CPA works on behalf of its user. The CPA checks each task requested by the user, discovers and invokes the corresponding cloud services or applications capable of executing that task, receives the results, stores them and, whenever the mobile client is available, it delivers them. In addition, the CPA controls the context processor, and manages the history data of both tasks and context.

The CPA achieves low energy usage by means of its mobile thin client architecture. By running a thin client, the demands on the mobile device itself are minimal. The thin client needs only pass minimal information for executing tasks to the CPA. Moreover, in the case of tasks previously run, information may not even have to be passed as the history stored by the CPA can be used. For some tasks, users may want to transfer a large amount of data to the CPA, such as uploading pictures, typically large in size, to cloud storage or various social networks. The thin client adapts intelligently so that if the user is on a cellular network, it will wait until connecting to a Wi-Fi network before uploading the pictures. This will save energy and reduce the cost for the user.

The CPA thin client uses a RESTful web service approach to communicate with the CPA middleware. RESTful approaches over HTTP are often more lightweight than more traditional SOAP based approaches as they can use JavaScript Object Notation (JSON), whereas SOAP uses the XML data format. JSON is smaller in size than XML, saving time and money during data transfer.

By moving more work to the cloud and reducing the workload on the mobile thin client, we have also saved energy in regards to some of the other approaches to energy saving in the mobile cloud area. For example, we do not need to waste energy by sending entire application copies or code to the cloud, and we can avoid the energy cost of profiling network conditions to determine if code offloading will actually result in energy savings.

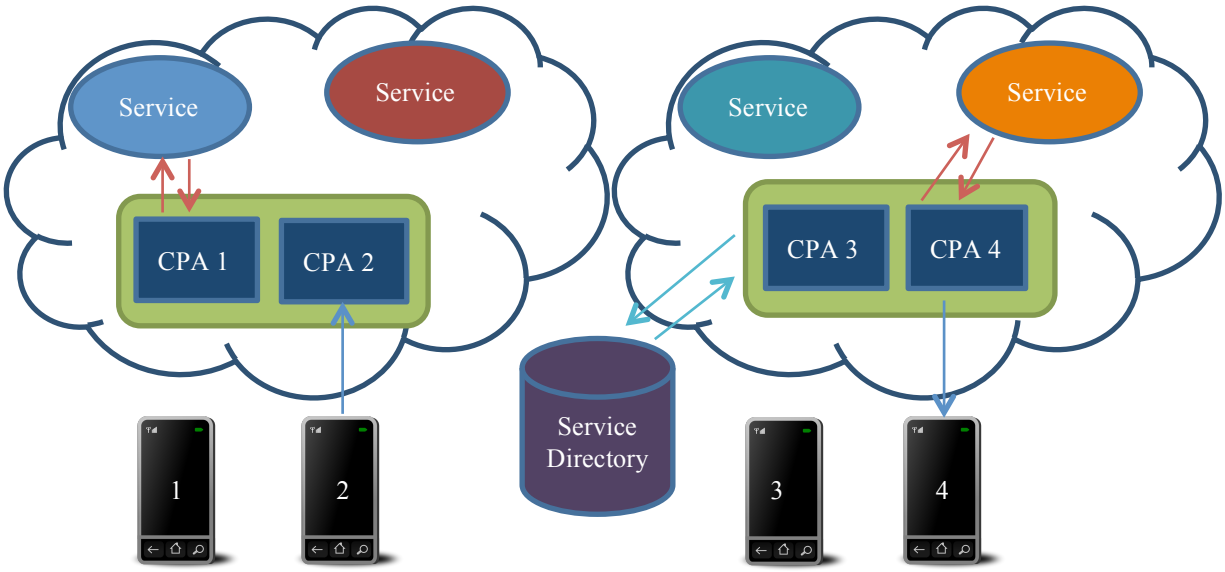

Figure 1: Mobile devices communicating with their CPA's in various clouds. The CPA's are located within the middleware deployments. Mobile 1 owns CPA 1, Mobile 2 owns CPA 2 etc. CPA 1 is invoking a service, disconnected from the mobile device. CPA 2 is receiving task data from the mobile device. CPA 3 is querying a service directory, disconnected from its mobile device. CPA 4 is receiving completed task information from its CPA, while invoking a service for another task.

The operation of the CPA is enhanced with the use of context. The mobile context is information about the current status of the mobile device, such as time of day, network connectivity, and sensor data. User context includes information about the current state of the user, such as social, work, and environmental status. The CPA middleware features a context processor, which can receive context data from the thin client. The context processor can then infer the context, and, when data is partial, by also using the history of past context(s).

Developers can create their own services, deploy them onto different clouds, and publish service descriptions to a central repository, which the CPA can use to find the appropriate service. The CPA can use preferences or constraints set by the user, such as constraints on time or cost, to determine which provider offering similar services it should choose from; the CPA can use these for independent operation during disconnection.

The CPA is capable of deployment in different clouds, and will move between them using secure web services. The thin client signals the middleware of the current mobile location, and the CPA moves to the closest cloud, carrying the data on currently executing tasks. The thin client is made aware of the new location. This brings the added benefit of minimising the round trip time between the mobile and the CPA, hosted in the cloud closest to the user.

There are two obvious benefits of this system:

- disconnection between the mobile client and the cloud computation handled by the CPA;
- energy savings resulting from minimum overhead, bandwidth use, and small latency when the CPA follows the mobile user by migrating between clouds.

3.3 Current Implementation

A Java-based prototype implementation of the CPA middleware was developed, deployed and tested on Amazon EC2. The Spring and Apache CXF frameworks were used to implement the services functionality. Apache jUDDI, a Java based implementation of a UDDI registry, was used as a service database. Sample cloud services were also developed, which include an Amazon S3 storage service, and an Amazon RDS database query service. These services were also deployed on EC2. The services themselves are described using the Web Services Description Language (WSDL). The WSDL files can be located anywhere on the web; the jUDDI database stores a URL reference to the file for each published service.

Mobile thin client applications have also been developed on the Android and iPhone platforms. These clients can take the details of tasks the user wishes to complete, and send them to their CPA in the cloud using RESTful web services. At this point, the thin clients can be shut down and the phone switched off; this will have no bearing on the task outcome.

Once the CPA has received the task information, it can then query the jUDDI database to discover an appropriate service, and then call that service, passing it the user data. If the task is a re-run of a task executed before using a previously discovered service, possibly with new task parameters, discovery is skipped, which significantly saves time. A re-run task only requires the user to send the ID of the task they wish to re-run to the CPA from the mobile device; they do not need to send all the parameters again, as they are already stored with the CPA. However if there are new parameters, they can be sent. This will cut the HTTP request size for re-run tasks. On a regular basis, it is more likely users will re-run variations of old tasks completed using previously discovered services, rather than execute new tasks requiring a new service to be discovered.

Currently, the implementation is limited to calling our sample services, as the used discovery mechanism facilitated by jUDDI is not very flexible, and does not provide for comparison or preference based search of services; it allows only search based on an approximate name match.

Once a service is completed and results returned to the CPA, the results are stored with the CPA in a database until the user is ready to receive them. An email notification is sent to the user, informing them of the ready results. Whenever the user is ready, s/he may open the thin client application on the mobile device, and view the results of the task. The operation of the CPA middleware is shown to be disconnected, asynchronous and energy efficient, compared to other approaches as previously discussed. The limitations of this approach that need to be addressed are the lack of standardisation amongst service descriptions for different types of services, and the lack of flexible discovery mechanisms.

The CPA is currently being extended to achieve the additional functionality discussed in this paper. A custom service database is currently being developed, which will allow greater flexibility in querying and comparing services, allowing use of the user preferences and constraints. The use of OWL-S, the web ontology language for web services, a standard adopted by the W3C, is being explored to describe services.

The context processor is also being actively developed. Context information can also be represented using OWL, which makes this representation a sensible choice for the middleware. Our mobile clients are also being extended to take this context information from the mobile clients.

The web services approach adopted not only for thin client to CPA communication, but for cloud based services, as discussed in the literature, also has the added benefit of being compatible with context data from mobile devices. This can also enable developers to implement mobile cloud services without any need for special considerations, aside from keeping the amount of data transferred to a sensible amount, with no unnecessary overhead.

4. Advantages

The main benefits of this system, disconnection and energy savings, are now explored.

4.1 Disconnected Operation

Over time, the user will switch between different Wi-Fi and cellular networks. Wi-Fi is rarely found in public. The coverage of 3G networks is not available everywhere, and even less so for 4G LTE networks, so they will become disconnected from both; neither can be relied upon. The disconnected mode of operation of the CPA is essential as when the mobile device loses a network connection, work can still occur. Once the CPA has been passed information on some task, it should independently complete it on behalf of the user. The CPA currently supports disconnected operation by:

- allowing the user to provide information for a new task, such as booking a flight or retrieving information from a large company database. This information is then sent to the CPA;
- allowing the user to signal the CPA to carry out a re-run task that has been performed previously, such as make a dinner reservation at a preferred restaurant or coordinate to meet a colleague at a suitable time, based on the current calendar. This only requires a task ID to be sent to the CPA; the details are already with the CPA.

Once the CPA has received the information, it will send an acknowledgement to the mobile device to indicate the task was received successfully. It will then work independently to discover a service capable of fulfilling the user's task, and then invoke that service with the information. Once the information has been received by the CPA, the thin client on the mobile device can shut down, the phone could become disconnected from the network, or the battery could die. None of these events will have any effect on the outcome of the task sent to the CPA, so we can see disconnected operation is possible here. This is a completely asynchronous approach.

When a task has completed execution, the result of the task, such as a booking confirmation number, or a database query result set, is stored with the CPA, and the user is informed that the result is now available. This can occur using Push notifications or email notifications. The user does not have to be immediately available to view the result. Whenever the user is ready to view the result, they can open the thin client application and choose to view the task result. At this point, the result information is downloaded to the device for viewing. If the user is currently using another mobile device, they could view the result on that as well by logging into their CPA with the thin client installed on that device.

It should also be clear from this approach, that once a task has been sent to the CPA, because of the disconnected and asynchronous nature, the mobile device has become freed up for other work.

Other approaches to mobile cloud do not gracefully handle disconnections. For example, if disconnection occurs when code offloading is taking place, the offloaded code must now be re-executed on the device after a timeout period. In the case of cloudlets, it would take a significant amount of time to save the changes to and move the VM-overlay back to the mobile device when the user moves away from the cloudlet. As we have seen, with the CPA, disconnection is not an issue. On the contrary, it is an operation mode.

4.2 Energy Conservation

The energy requirements of mobile cloud computing on the mobile device are not insignificant. Typically, due to the high latency and low bandwidth on wide area cellular networks such as 3G, the energy usage is higher than when a Wi-Fi network is used. This does not just apply to mobile cloud computing, but to all client-server applications running on mobile devices. This is a growing problem as highlighted in [11]. As users run more applications on their mobile devices that connect with online services, and more data is transferred over edge networks from the mobile device to the cloud networks, the edge traffic is dominating the energy usage in the access of cloud services. Therefore, energy conservation in mobile cloud computing is an essential consideration and, when achieved, is a benefit as energy supplies are at a premium.

We have already described the optimisations used by the thin client. We make a distinction between new tasks and re-runs of previous tasks, which require less data transfer to and from the mobile device. Secondly, the RESTful approach with the JSON message format for communicating with the CPA middleware will result in fewer characters needed to be sent and received over the network in the HTTP message payload. The result is energy savings from the network interface controller and the time spent transferring and processing the data. We also mentioned cost savings, as fewer data characters are being sent over the cellular network, there is a smaller charge for the user.

5. Application scenarios

The driving factor behind the implementation of the CPA is its disconnected operation mode, asynchronous nature and energy saving. The general public will interact with cloud-based services by delegating work to the CPAs from the mobile devices.

Consider a company's private cloud that the CPA has been authorised to operate with, as the owner is an employee of that company. The CPA will have access to private services related to the specific operation of that company. It can either be loaded with service information when it is first granted access, and can consult private service databases. This will enable cooperation between different departments and areas of the company. A sales representative out on a client visit will be able to signal the CPA to retrieve the most recent prices of wholesale items from previous visits to that employee. The CPA can determine which client the user is with based on the location context, and will know from task history

which items to check the current prices of based on past orders. The CPA uses the company's private cloud services to fetch the information using the task and history context to query the correct information.

The ability of the CPA to move between different clouds enables strong synchronisation of data between them. A user who wants to synchronise photos between different social networks, such as Facebook and Twitter, can simply choose the pictures s/he wants to upload from within the gallery application on their mobile device and select to upload the photos from the mobile to the CPA. The user can specify at this time which networks they would like the photographs synchronised to. At this point the thin client can check if the user is connected to a Wi-Fi or 3G network, and choose to either go ahead with the upload or queue the operation respectively. If it will queue the operation, the thin client will wait until the device is connected to a Wi-Fi network (this policy of course can be overridden by the user). Once the pictures are uploaded to the CPA, which knows the details of and is authorised to access the different social network accounts, it can then post the pictures to each social network. The user does not need to waste bandwidth, money, time and energy uploading the same pictures individually from the mobile device to each network.

The CPA middleware can support smart city applications in the areas of utilities optimal management, traffic, health, education, commerce and entertainment for all its inhabitants. Mobile devices equipped with sensors such as the camera, microphone and GPS as well as sensor networks deployed in different areas of cities will upload their data to the CPA. Cloud services will refine sensor data from CPAs and produce information such as current traffic, pollution map, habitual patterns of movement and access to city utilities and services. As a scenario, consider thousands of mobile phones that while on the move (with their users) send street noise data sampled by the microphone together with location (GPS) by the Wi-Fi access points to the cloud, at different time of the day. In addition to air pollution data sent by sensor networks, these data can be used to create a map of the city environment quality. The same devices can help determine people movement patterns at different time of the day leading to decisions regarding street/transportation improvements, temporary services offered to pedestrians or drivers, etc. Not only local authorities will benefit of this system, but city inhabitants will use their mobiles to access information (to which they contributed) as a new city utility. They can be incentivised to volunteer their devices by getting free access to the new services.

Consider a final, more complex scenario, where the user would like to book a flight to San Francisco. The battery on the mobile device is low, and if the web browser is opened and used to book tickets on the website, the battery may die, so the CPA is used instead. The user has used the CPA to book flights before, but not to San Francisco. The CPA is signalled to re-run the flight booking task, but changes the destination to San Francisco. The CPA can use the current location context from the GPS functionality on the device to determine that the flight should leave from Dublin, where the user lives (this could be overridden if it were not the case). The task data is sent to the CPA from the thin client which includes the ID of the flight booking task run before, and the new departure and arrival locations. The thin client receives an acknowledgment response. The CPA uses the task history and knows that the user previously set a constraint that the cost of the flights should not be more than €1500. The task history also indicates that Aer Lingus was previously used for a flight, so the CPA contacts the Aer Lingus booking API to book the flight, which may be a SOAP or RESTful web service. Unfortunately, Aer Lingus no longer flies directly to San Francisco from Dublin. The API recommends flying from Dublin to New York JFK with Aer Lingus, and then connecting to a codeshare partner, JetBlue, to fly from JFK to San Francisco. Unfortunately, the total price of the journey is €1700, and this violates the constraint on the price, so the CPA rejects the offer. The CPA instead consults another airline booking API from the task history, British

Airways. British Airways recommends flying to London Heathrow, and connecting to another British Airways flight to San Francisco. The cost comes in at €1400. This is within the constraint value, so the CPA makes the booking. The confirmation number and receipt is sent back to the CPA for the user to view later on the thin client application, possibly when they get home and charge the battery. The CPA can then also recommend to book hotels in San Francisco, within price ranges previously used for hotel bookings. As the previous travel context history shows, the user has used the CPA to book hotels at flight destinations as well.

6. Conclusions

Mobile cloud computing is a new paradigm for running mobile applications on cloud infrastructure. It can expand the range of tasks a mobile device can be used for by invoking existing and new cloud services, such as storage and computation facilities. The merging of these two platforms is not without implementation difficulties, and several approaches to this were surveyed in this paper. None of these approaches have considered the overall user experience of mobile cloud applications as a design requirement. Our approach, the cloud personal assistant middleware, takes the user experience as the primary goal of design. CPA introduces disconnected and asynchronous operation in the mobile cloud ("more than a cloud"), as well as saving energy. We studied use cases that the CPA can enable and potential applications. The current implementation was described as well as new functional additions. While still in development, the CPA middleware aims to become a commercial product.

Acknowledgement

The PhD research of Michael J. O'Sullivan is funded by the Embark Initiative of the Irish Research Council.

References

[1] CISCO IBSG, "The Road to 'Cloud Nine'. How Service Providers Can Monetize Consumer Mobile Cloud", White paper, Febr 2013, http://www.cisco.com/web/about/ac79/docs/sp/Mobile-Cloud.pdf

[2] ABI Research, "Mobile Cloud Computing Subscribers to Total Nearly One Billion by 2014", http://www.abiresearch.com/press/1484-Mobile+Cloud+Computing+Subscribers+to+Total+Nearly+One+Billion+by+2014.

[3] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing", IEEE Pervasive Computing, 2009; 8(4), pp. 14-23.

[4] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: Enabling mobile phones as interfaces to cloud applications", Middleware 2009, pp. 83-102.

[5] P. Simoens, F. De Turck, B. Dhoedt, and P. Demeester, "Remote Display Solutions for Mobile Cloud Computing", Computer, 2011, 44(8), pp. 46-53.

[6] E.Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, et al. "MAUI: making smartphones last longer with code offload", Proceedings of the 8th international conference on Mobile systems, applications, and services, San Francisco, California, USA. 1814441, ACM, 2010, pp. 49-62.

[7] B-.G. Chun, S. Ihm, P. Maniatis, M. Naik and A. Patti, "CloneCloud: elastic execution between mobile device and cloud", Proceedings of the sixth conference on Computer systems, Salzburg, Austria, 1966473, ACM, 2011, pp. 301-314.

[8] D. Kovachev, Y. Tian and R. Klamma, "Adaptive Computation Offloading from Mobile Devices into the Cloud", 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2012, pp. 784-791.

[9] R. K. K. Ma, L. King Tin and W. Cho-Li, "eXCloud: Transparent runtime support for scaling mobile applications in cloud", 2011 International Conference on Cloud and Service Computing, 2011, pp. 103-110.

[10] M. J. O'Sullivan, D. Grigoras. The Cloud Personal Assistant for Providing Services to Mobile Clients, IEEE MobileCloud, Redwood City, San Francisco Bay, USA, 2013, pp. 477-484.

[11] Centre for Energy-Efficient Telecommunication, Bell Labs and University of Melbourne Report, "The Power of Wireless Cloud", http://www.ceet.unimelb.edu.au/pdfs/ceet_white_paper_wireless_cloud.pdf, April 2013.