The Journal of Systems and Software 92 (2014) 82–94

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

# Mobile Cloud Middleware

Huber Flores*, Satish Narayana Srirama

*University of Tartu, Institute of Computer Science, Mobile Cloud Lab., J. Liivi 2, Tartu, Estonia*

ARTICLE INFO

*Article history:*
Received 30 September 2012
Received in revised form 9 September 2013
Accepted 9 September 2013
Available online 23 September 2013

*Keywords:*
Mobile Cloud Computing
Task delegation
Code offloading
Interoperability

ABSTRACT

Mobile Cloud Computing (MCC) is arising as a prominent research area that is seeking to bring the massive advantages of the cloud to the constrained smartphones. Mobile devices are looking towards cloud-aware techniques, driven by their growing interest to provide ubiquitous PC-like functionality to mobile users. These functionalities mainly target at increasing storage and computational capabilities. Smartphones may integrate those functionalities from different cloud levels, in a service oriented manner within the mobile applications, so that a mobile task can be delegated by direct invocation of a service. However, developing these kind of mobile cloud applications requires to integrate and consider multiple aspects of the clouds, such as resource-intensive processing, programmatically provisioning of resources (Web APIs) and cloud intercommunication. To overcome these issues, we have developed a Mobile Cloud Middleware (MCM) framework, which addresses the issues of interoperability across multiple clouds, asynchronous delegation of mobile tasks and dynamic allocation of cloud infrastructure. MCM also fosters the integration and orchestration of mobile tasks delegated with minimal data transfer. A prototype of MCM is developed and several applications are demonstrated in different domains. To verify the scalability of MCM, load tests are also performed on the hybrid cloud resources. The detailed performance analysis of the middleware framework shows that MCM improves the quality of service for mobiles and helps in maintaining soft-real time responses for mobile cloud applications.



## 1. Introduction

Mobile Cloud Computing (MCC) is arising as a prominent research area that is seeking to bring the massive advantages of the cloud to the constrained smartphones and to enhance the telecommunication insfrastructures with self-adaptive behavior for the provisioning of scalable mobile cloud services. MCC focuses on the benefits that can be achieved by the mobile resources when a mobile operation such as data storage or processing is delegated or offloaded to the cloud (Satyanarayanan et al., 2009; Flores et al., 2011; Flores and Srirama, 2013). These benefits include extended battery lifetime, improved storage capacity and increased processing power, thus enriching the mobile applications along with the mobile user experience. Moreover, MCC focuses on finding an optimal configuration of a mobile cloud infrastructure in order to handle the oscillating telecommunication loads (scale-out), to facilitate the process of deploying services without managing the underlying technology (on the fly) and to reduce operational and provisioning costs (pay-as-you-go model) (Flores and Srirama, 2012).

Mobile cloud applications (Flores et al., 2012) are considered as the next generation of mobile applications, due to their promise of bonded cloud functionality that augment processing capabilities on demand, power-aware decision mechanisms that allow to utilize efficiently the resources of the device, and their dynamic resource allocation approaches that allow to program and utilize cloud services at different levels (SaaS, IaaS, PaaS). However, adapting the cloud paradigm for mobile devices is still in its infancy and several issues are yet to be answered. Some of the prominent questions are; how to decide from the smartphone, the deployment aspects (e.g. type of instance) of a mobile task delegated to the cloud? How to decrease the effort and complexity of developing a mobile application that requires accessing distributed hybrid cloud architectures? How to handle a multi-cloud operation without overloading the mobile resources? How to keep the properties (e.g. memory use, application size, etc.) of a mobile cloud application similar to that of a native one?

Hybrid cloud and cloud interoperability are essential for mobile scenarios in order to foster the de-coupling of the handset to a specific cloud vendor, to enrich the mobile applications with the variety of cloud services provided on the Web and to create new business opportunities and alliances (Zeng et al., 2004). However, developing a mobile cloud application involves adapting different Web APIs from different cloud vendors within a native mobile platform. Vendors generally offer the Web API as an interface that

* Corresponding author. Tel.: +372 56090142.
*E-mail addresses:* huber@ut.ee (H. Flores), srirama@ut.ee (S.N. Srirama).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2013.09.012

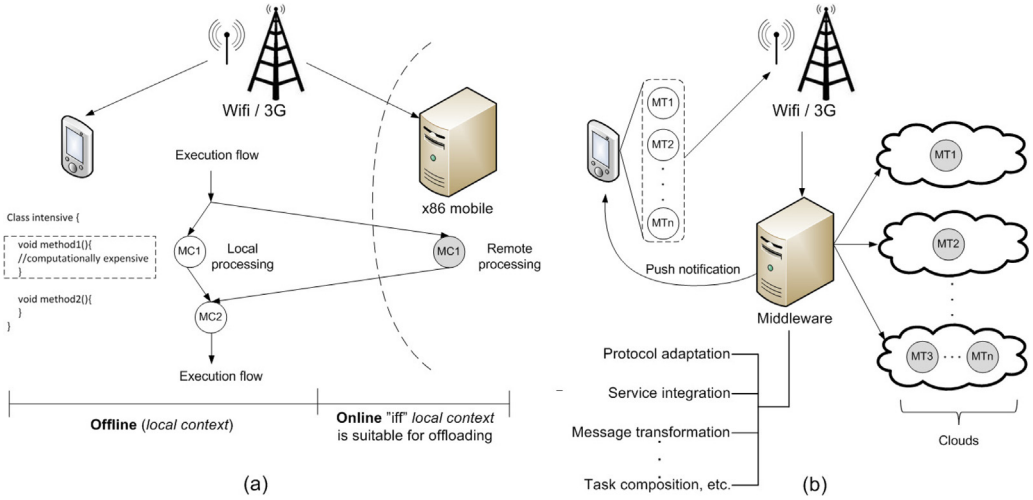


**Fig. 1.** (a) Code offloading architecture and (b) task delegation architecture.

allows programming the dynamic computational infrastructure that support massively parallel computing. Deploying a Web API on a handset is demanding for the mobile operating system due to many reasons like compiler limitations, additional dependencies, code incompatibility etc., and thus in most of the cases the deployment just fails. Moreover, adapting a Web API requires specialized knowledge of low level programming techniques and most of the solutions are implemented as ad-hoc.

In terms of data storage facilitation of the cloud, existing mobile cloud approaches such as data synchronization (via SyncML) allow the deployment of a Web API on the device for retrieving data from the cloud. For instance, Funambol (Onetti and Capobianco, 2005) or gdata-calendar Web API can be integrated to an Android application to synchronize calendar information (e.g. alarms, tasks, etc.). However, this approach focuses on replicating the data located in the cloud with the data located in the handset, which is not a real improvement. Alternatively, cloud services can be encapsulated as Web services that can be invoked directly using a simple REST mobile client. However, due to the time consuming nature of a cloud request, this can cause an overhead in the mobile resources, without a proper asynchronous communication mechanism. Moreover, by using a REST mobile client, the device is forced to perform multiple transactions and to handle the results of those transactions locally, which is costly from the energy point of view of the handset. The number of transactions are directly associated with the number of cloud services utilized in the mobile cloud application.

To counter the problems such as the interoperability across multiple clouds, invocation of data-intensive processing from the handset, dynamic configuration of execution properties of a delegated task in the cloud, and to introduce the platform independence feature for the mobile cloud applications, we propose the Mobile Cloud Middleware (MCM) (Flores et al., 2011). The middleware abstracts the Web API of different/multiple cloud levels and provides a unique interface that responds (JSON-based) according to the cloud services requested (REST-based). MCM provides multiple internal components and adapters, which manage the connection and communication between different clouds. Since most of the cloud services require significant time to process the request; it is logical to have asynchronous invocation of the cloud service. Asynchronicity is added to the MCM by using push notification services provided by different mobile application platforms and by extending the capabilities of a XMPP-based IM infrastructure (Flores and Srirama, 2013).

Furthermore, MCM fosters a flexible and scalable approach to create hybrid cloud mobile applications based on declarative service composition. Service composition is considered for representing each mobile task to be delegated as a MCM delegation component that is depicted graphically in an Eclipse plugin so that the cloud services that conform a mobile cloud application can be modeled as a data-flow structure based on user driven specifications. Once developed, a composed task is deployed within the middleware for execution that is triggered by a single invocation from the mobile. Finally, MCM prototype was extensively analyzed for its performance and scalability under heavy loads, and the details are addressed in further sections.

The rest of this paper is organized as follows: Section 2 addresses the related work and highlights the issues and challenges that need to be investigated in order to implement a mobile cloud architecture based on delegation. Section 3 introduces the concept of Mobile Cloud Middleware. Section 4 discusses the MCM hybrid cloud composition mechanism. Section 5 presents a scalability analysis of the framework and Section 6 concludes the paper with future research directions.

## 2. Related work and state of the art

Recently, there has been a growing interest to bind cloud resources to low-power devices such as smartphones in order to provide PC-like functionality to the mobile users (Balan et al., 2002). This loose integration happens through a mediator (aka middleware) that controls every aspect in the communication and coordinates the interaction (online/offline) between the back-end infrastructure and the mobile device. Currently, two main middleware criterias are utilized to bring the cloud to the vicinity of a mobile; offloading and delegation. The architecture for each approach is shown in Fig. 1.

In a delegation model, a mobile device utilizes the cloud in a service oriented manner in order to integrate services running at different cloud levels within the mobile applications so that a mobile task (MT) can be delegated by invoking a cloud service from the handset. This kind of approach requires to have always available network connectivity and conceptually a mobile task is delegated when it is computationally unfeasible for the mobile resources (Flores et al., 2011). In contrast, in an offloading model, a mobile application may be partitioned (e.g. methods, classes) and analyzed *a priori* (at development stages) or *a posteriori* (at runtime) so that the most computational expensive operations (aka mobile components) at code level can be identified and offloaded for remote processing (Gu et al., 2004; Li et al., 2001). A mobile component (MC) may be offloaded or not, depending on the impact of