UNIVERSITY OF TARTU
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
Institute of Computer Science

Huber Flores

Mobile Cloud Middleware

Master Thesis (30 EAP)

Supervisor: Satish Srirama, PhD

Author:..... ".....” May 2011

Supervisor:..... ".....” May 2011

Professor:..... ".....” May 2011

TARTU, 2011

Abstract

Both cloud computing and mobile computing domains have advanced rapidly and are the promising technologies for the near future. Cloud computing becomes mobile when a mobile device tries to access the shared pool of computing resources provided by the cloud, on demand. Mobile technologies are mainly drawing their attention to the clouds due to the increasing demand of the applications, for processing power, storage space and energy. Mobile applications take advantage of the elasticity of the cloud to remotely process the task but at the same time maintain a soft-real time response behavior.

However, developing mobile cloud applications is tricky and involves working with services and APIs from different cloud vendors. Most often these APIs are not interoperable and the information which is processed and stored into the cloud is not transferable to other clouds. To counter the problems the thesis studied a middleware solution, Mobile Cloud Middleware (MCM), which handles the interoperability issues, and eases the use of processing intensive services from mobile phones. A prototype of MCM was developed and the detailed performance analysis of the framework shows that the middleware framework improves the quality of service for mobiles and helps in maintaining soft-real time responses.

The thesis also has realized several case studies using the MCM. One of them is Zompopo, an Android application that provides an intelligent calendar, combining Google Calendar and the accelerometer sensor of the mobile, which allows the user to schedule his/her activities from the beginning of the day according to his previous week's activities. The application is explained with detailed architectural and technological choices. The application uses MapReduce to analyze the accelerometer sensor data to deduce

any diversions from the regular calendar activity, thus efficiently utilizing the cloud computing resources. A detailed performance analysis of the application is also provided, showing how mobile application benefit by going cloud-aware.

Apart from Zompopo, the thesis also tried to realize some social network applications using the MCM. Social networks have become quite popular these days and the creation of social groups of people with common interests, results in sharing and collaborative relationships among the members. CroudSTag, the developed Android application, aids in forming social groups of common interest, from the mobile devices. The application obtains a set of pictures from a storage cloud, uses the face recognition cloud services to identify the people, and forms social groups on Facebook, a well known social network. The application is explained with detailed architectural and technological choices. The performance analysis of the application shows that the social groups can be formed with significant ease and reasonable performance latencies from the mobile devices with the help of MCM.

Contents

List of Figures

LIST OF FIGURES

1

Introduction

1.1 Introduction

Both cloud computing (1) and mobile computing domains have advanced rapidly and are the promising technologies for the near future. Cloud computing is a style of computing in which, typically, resources scalable on demand are provided "as a service (aaS)" over the Internet to users who need not have knowledge of, expertise in, or control over the cloud infrastructure that supports them. The provisioning of cloud services can occur at the Infrastructural level (IaaS) or Platform level (PaaS) or at the Software level (SaaS). Cloud computing mainly forwards the utility computing model, where consumers pay on the basis of their usage. Moreover, cloud computing promises the availability of virtually infinite resources.

On the other hand, improvements in mobile devices, on hardware (embedded sensors, memory, power consumption, touchscreen, better ergonomic design, etc.), in software (more numerous and more sophisticated applications due to the release of iPhone (2) and Android (3) platforms) and in transmission (higher data transmission rates achieved with 3G and 4G technologies), have contributed towards having higher mobile penetration and better services provided to the customers. Now the mobile users can collaborate and share information with significant ease.

Mobiles enter cloud computing domain by trying to access the shared pool of computing resources provided by the cloud on demand. Clouds are looking forward to the mobile domain, having their expectations focused in the idea of data synchronization services. Mobile sync refers to the synchronization of data in the handset with a server

and a portal in the cloud. However, Mobile technologies are drawing the attention to the clouds due to the demand of the applications, for processing power, storage space and energy saving. This has led to the appearance of the Mobile Cloud Computing (MCC) domain. Applications benefiting from a Mobile Cloud are related to various domains such as social networks, location based services, context-aware systems etc.

1.1.1 Motivation

The proliferation of mobile devices is fostering the emergence of ubiquitous environments, and thus the development of pervasive and context-aware applications is increasing. Examples include, AAMPL (4), MetroSense (5), Place-Its (6) etc. Context-aware applications are those that utilize the user's context for providing environmental adaptability. These kinds of applications enable user interaction by combining the functionality of a mobile application with a variety of sensors (accelerometer, magnetic field, gravity, etc.) within the handset and with external entities as location information systems (GPS). Sensor's information allows the applications to learn from the user in order to provide better services. However, the use of sensors in the applications is limited due the lack of resources in the handset for storing and processing the data collected.

Furthermore, a next generation of mobile applications is focusing in the combination of different cloud capabilities from multiple clouds for the creation of powerful mashups that are not tied to cloud specifications, and thus can be migrated easily to another cloud solutions.

1.1.2 Contributions

To counter the problems with the interoperability across multiple clouds, to perform data-intensive processing invocation from the handset and to introduce the platform independence feature for the mobile cloud applications, the following thesis discusses a Mobile Cloud Middleware (MCM). The middleware provides a unique interface for mobile connection and multiple internal interfaces and adapters, which manage the connection and communication between different clouds. The MCM capabilities for managing resource intensive tasks can easily be envisioned in several scenarios. One of them includes the provisioning of context-aware services for processing data collected by the accelerometer with the purpose of creating an intelligent calendar that

predicts human activities. Another scenario consist in the formation of a social group by recognizing people in a set of pictures stored in the cloud.

Since, most of the reasonable mobile cloud services require significant time to process the request; it is logical to have asynchronous invocation of the mobile cloud services. Asynchronicity is added to the MCM by using push notification services provided by different mobile application development platforms. Once the MCM prototype was developed, it was extensively analyzed for its performance and the detailed are addressed here. MCM also improves the quality of service (QoS) for mobiles and helps in maintaining soft-real time responses. Moreover, MCM implements an interface with Funambol (7) to monitoring from the mobile the status of the task.

1.1.3 Outline

Chapter 2: discusses the state of the art addressed by this thesis. The chapter first introduces cloud computing technologies along with the cloud services available though the Web for fitting the mobile demand. Later introduces the developments in mobile technologies and how those are taking advantage of the cloud using the synchronization data approach. The chapter concludes by making emphasis in current mobile cloud providers

Chapter 3: explains the problems regarding the combination and the invocation of cloud services from the mobile phone. First it introduces the main problem regarding with cloud interoperability. Second, it covers cloud resource dependency according to the mobile platform. Third, it introduces the problem of invoking data-intensive resources from the handset

Chapter 4: addresses the mobile cloud middleware concept with realization details. The chapter first introduces the concept of pure mobile cloud services. Later, explains how applications can benefit from the cloud invoking the storage and the computational services. Further, introduces the asynchronous notification feature for managing cloud services. it concludes the chapter by explaining how MCM can be integrated with existent solution such as Funambol.

Chapter 5: introduces two applications (Zompopo and CroudSTag) as case of study for MCM. Each application consist of a description, development explanation and performance analysis. First is introduces Zompopo as a context-aware application that can relies in the provisioning of context-aware services from the cloud. Second, it explains CroudSTag as a social application that relies on cloud services for the performed face recognition in a set of pictures stored at the cloud

Chapter 6: provides the conclusion about the findings of the thesis.

Chapter 7: describes the future research directions such the integration with the Mobile Web Services Mediation Framework (MWSMF (8)) and scalability improvements such the re-implementation of the middleware solution using Erlang (9).

2

State of the Art

The state of the art used in the thesis highlighted the advances in the cloud computing domain and the mobile domain. Cloud computing is changing the style in which services are delivered over the Internet. A cloud service is able to supply scalability according to the demand, to provide functionality without dealing the underlying technologies that conform the cloud, and to leverage the distributed physical infrastructure for using the cloud as a virtual pool of resources (pay as you need). The consumption of cloud services for mobile devices is focused in the synchronization of data (Mobile sync). However, Most of the cloud services available from the Web are extending to mobile version for fitting the demand of ubiquity access.

The integration of the cloud with mobile technologies has envisioned the Mobile Cloud Computing domain (MCC). This chapter introduces these developments from the literature and addresses the motivation for the research in the MCC domain.

2.1 Cloud Computing

Cloud computing is a style of computing in which, typically, resources scalable on demand are provided "as a service (aaS)" over the Internet to users who need not have knowledge of, expertise in, or control over the cloud infrastructure that supports them. Cloud computing mainly forwards the idea of utility computing along with virtualization (10). In the utility computing model, consumers pay based on their usage of computing resources, just like the traditional utility services e.g. water, electricity, gas etc. Just like any utility services model cloud computing benefits from economies

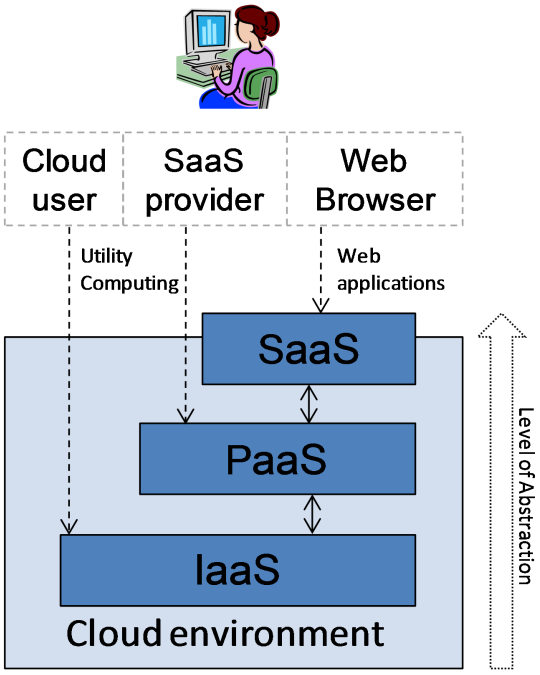

Figure 2.1: Level of abstraction and layers of cloud services

of scale. On the other hand, virtualization technologies partition hardware and thus provide flexible and scalable computing platforms. Servers in the cloud can be physical machines or virtual machines. A cloud computing platform dynamically provisions, configures, reconfigures, and de-provisions servers as requested.

Cloud services are provided on demand and at different levels. Figure 2.4 shows the layers of cloud services, in terms of level of abstraction. The provisioning of services can be at the Infrastructural level (IaaS) or Platform level (PaaS) or at the Software level (SaaS). In the IaaS, commodity computers, distributed across Internet, are used to perform parallel processing, distributed storage, indexing and mining of data. IaaS provides complete control over the operating system and the clients benefit from the computing resources like processing power and storage, e.g. Amazon EC2 (11). Virtualization is the key technology behind realization of these services. PaaS mainly provides hosting environments for other applications. Clients can deploy the domain specific applications on these platforms, e.g. Google App Engine (12). These applications are in turn provided to the users as SaaS. SaaS are generally accessible from web browsers, e.g. Facebook. Web 2.0 is the main technology behind the realization

of SaaS. However, the abstraction between the layers is not concrete and several of the examples can be argued for other layers.

While there are several public clouds on the market, Google Apps (Google Mail, Docs, Sites, Calendar, etc), Google App Engine (provides elastic platform for Java and Python applications with some limitations) and Amazon EC2 are probably most known and widely used. Elastic Java Virtual Machine on Google App Engine allows developers to concentrate on creating functionality rather than bother about maintenance and system setup. Such sandboxing, however, places some restrictions on the allowed functionality. Amazon EC2 on the other hand allows full control over virtual machine, starting from the operating system. It is possible to select a suitable operating system, and platform (32 and 64 bit) from many available Amazon Machine Images (AMI) and several possible virtual machines, which differ in CPU power, memory and disk space. This functionality allows to freely select suitable technologies for any particular task. In case of EC2, price for the service depends on machine size, its uptime, and used bandwidth in and out of the cloud.

Moreover, there are free implementations of EC2 compatible cloud infrastructure e.g. Eucalyptus (13), that help in creating private clouds. Thus the cloud computing applications can initially be developed at the private clouds and later can be scaled to the public clouds. The setup is of great help for the research and academic communities, as the initial expenses of experiments can be reduced by great extent. Our research group is in the process of setting up a scientific computing cloud (SciCloud) on our clusters, using Eucalyptus technology. With this SciCloud (14), students and researchers can efficiently use the already existing resources of university computer networks, in solving computationally intensive scientific, mathematical, and academic problems. The project mainly targets the development of a framework, including models and methods for establishment, proper selection, state management (managing running state and data), auto scaling and interoperability of the private clouds. The preliminary results can be found at the project site

2.1.0.1 Amazon

Amazon (11) is a public cloud computing provider that offers services (AWS) based in the IaaS approach. Amazon AWS (Amazon Web Services) is a set of Web Services (WS) (15) that relies on the top of the cloud computing infrastructure for delivering

services that can be accessed using REST (Representational State Transfer) and SOAP (Simple Object Access Protocol) protocols. Within the bundle of services provided in the Amazon stack, EC2 (Elastic Compute Cloud) and S3 (Simple Storage Service) can be mentioned since are the most popular and well-known services. Other services have been developed around these basic services such EBS (Amazon Elastic Block Store), AWS Management Console, etc. Moreover, one of the latest services provided by Amazon consist in CloudWatch for monitoring the applications that are running in the cloud.

Amazon services are paid according to the user's consumption (number of requests, amount of bandwidth, etc). However, in February (2011), Amazon just released a free tier account for the developers in order to foster the creation of applications based on their cloud infrastructure. In the context of mobile technologies, Amazon provides support for Android. Although support still in beta version.

2.1.0.2 Eucalyptus

Eucalyptus is a solution for the implementation of private clouds. It offers an enterprise solution for business in general and a open source version for the open source community. Eucalyptus architecture is based on Amazon, and thus they share similar composition for the provisioning of services. Walrus provides the support for storage (like S3) and Eucalyptus provides the computational service (like EC2). However, applications that are developed for Eucalyptus are not highly compatible with Amazon. Therefore, applications must suffer some changes when is necessary to migrate from one architecture to another.

2.1.0.3 Google Application Engine

The Application Engine contains all the set of services provided by Google. it uses a SaaS approach for delivering services over the Internet. Among the most well-known services we can mention, Google Analytics, Google docs, Picasa, etc. it provides also support for storage (Google for developers).

Android mobile platform is tied to the solutions provided by Google, and thus most of the services released over the Internet were extending for proving a mobile version. As an example, Android set of applications (Calendar, e-mail, contact, etc).

can be synchronized easily (Google Calendar, Google sync, etc) if the user has a Google Account.

2.2 Mobile Technologies

Improvements in mobile devices, on hardware (embedded sensors, memory, power consumption, touchscreen, better ergonomic design, etc.), in software (more numerous and more sophisticated applications due to the release of iPhone and Android platforms) and in transmission (higher data transmission rates achieved with 3G and 4G technologies), have contributed towards having higher mobile penetration and better services provided to the customers. Also, those improvements have enabled the mobiles to become a source of information, to understand the user in multiple ways (interaction, movement, location etc.). Smart phones are rich of a variety of sensors that enable sensing the user environment for providing adaptability in real-time.

Nowadays, a set of defaults sensors is embedded within the smartphones. Most of the sensors are highly integrated with the software and hardware for providing multiple functions related with usability, security, etc. The accelerometer is the most common sensor within the mobile phone that is used for the sensing the acceleration of the device in different axes (generally 3). The accelerometer can be used for providing multiple services, for example, vendors are using the accelerometer as a security mechanism for securing the hard disk when the device experiences a free fall. The study has used the sensor in one of the applications, Zompopo, which is explained in detail in section 5.1. Similarly to accelerometer, the magnetic field sensor can also sense movement. It is able to sense changes in the magnetic field of the objects. Since the magnetic field shows variation that are related with movements, this kind of sensor may be used for collecting precise information such distance, direction etc.

Apart from the sensors, mobile devices also come with other accessories like the built in camera, support for GPS etc. Those hardware is use for collecting information that may be used for performing tasks in real-time. However, some of this information demands high processing, and thus the use of this information is passive from the mobile perspective.

2.2.1 Mobile Platforms

Mobile applications are developed for a diversity of mobile platform including Symbian (16) (Nokia), Windows Mobile (17) (Microsoft), Blackberry (18) (Sony Ericsson) etc. Recently, Symbian have been replaced from the Market for Windows Mobile since the popularity of this decrease in the mobile market. Since the introduction of iOs (2007) and Android (2008) as mobile platforms for the smartphones; a huge popularity has been growing around these two operating systems as basis for the development of mobile applications. Apple released iOs as platform for their own device (iPhone). Similary, Android was released as open source in the consortium of the Open Handset Alliance. Currently, Android is supported by vendors such as Samsung, Sony Ericsson, HTC, Toshiba, LG among others.

2.2.1.1 Android Platform

Its a mobile platform released by the Open Hanset Alliance that consist in a software stack composed by an operating system, middleware and key applications. The development of software is tied to the use of the Delvik Virtual Machine (Android Runtime) that enables the use of Java as programming language. Most of the libraries which are compatible with a JDK, can be deployed within Android. However, Some of them may present issues concerning compatibility with the compiler and thus, are unable to be executed. For example typica (19) API presents such integration problems.

Android architecture consists of multiple layers that relies on a Linux kernel as shown in figure 2.2. The application layer includes a set default applications within the operating system such calendar, contacts, etc. Those applications can be synchronized with the cloud using Google sync. The application framework consist in predetermined services for managing device resources as hardware (sensors, screen, etc), software (alarms, background services) and the integration with external sources (location information systems, AC2DM notification service, etc). Within the set of libraries, Android incorporate those that provide real-time performance. For example a light version of SQLite database is include.

The distribution of applications for Android is through the Android market. Although, applications can be distributed freely over the Internet once they are packed in APK (Android Package Files) files. However, if the application it was not purchased

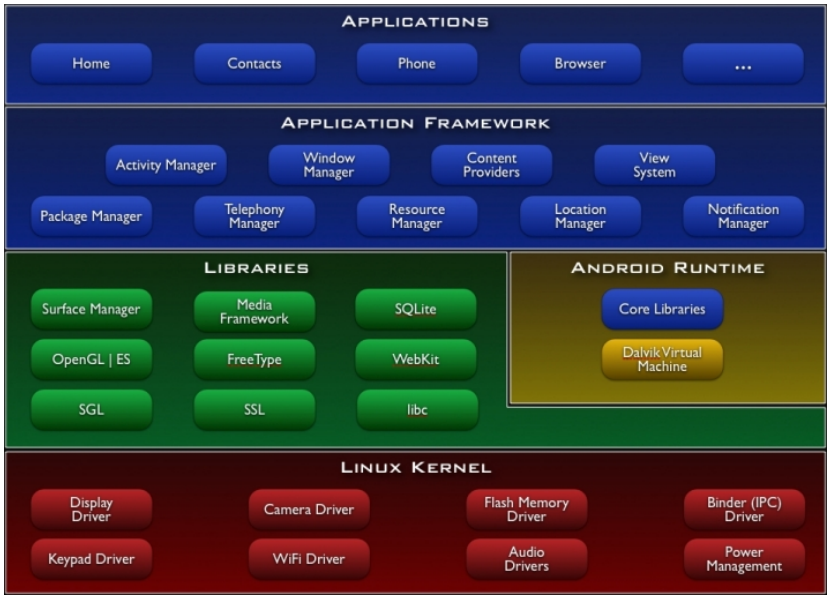

Figure 2.2: Android architecture

in the Android Market, still it may be installed, but under the risk of acquire Android Malware.

2.2.1.2 iOS Platform

its a mobile platform owned by Apple, Inc. that's its deployed for its mobile device (iPhone). Since its a proprietary technology most of its core functionality altogether with the hardware is not accessible for the developer. iOS consists of a number of multiple software layers, each layer allows the use of programming frameworks for the development of applications that run on top of the underlying device architecture.

iOS architecture is depicted in figure 2.3. The conception of each layer allows multiple levels of abstractions for dealing with the hardware. The complexity of each layer is related with the lines of code needed for achieve the developer objective in the mobile application. In general, the higher level of layer, the less effort required for building the application.

Cocoa Toach Layer is the higher layer of the iOs platform that is based in c-objective language, it provides services such Push Notification Service, Game Kit Frameworks, among others. Media Layer provides capabilities for reproducing video, audio altogether with graphics capabilities for animations. The core services layer contains another

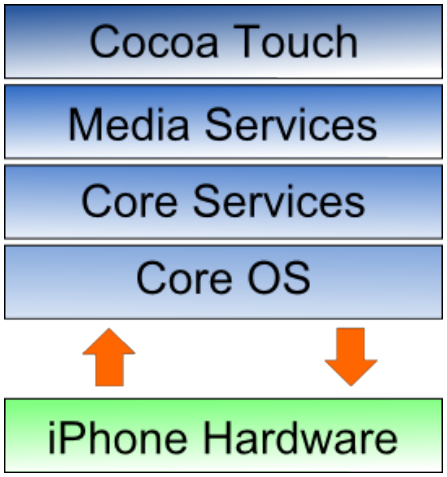

Figure 2.3: iOS architecture

abstraction the services described in Cocoa Toach and Media Services. The Core OS Layer lies in the bottom position of the iOS stack and, as such, sits directly on top of the device hardware. The layer provides a variety of services including low level networking, access to external accessories and the usual fundamental operating system services such as memory management, file system handling and threads.

The distribution of application is exclusively through the iPhone App Store. A developer that desires publish an applications must submitted first for inspection in the iOS Dev Center. Once, the review is done and the application fits the mandatory requirements requested by Apple, just then the application is published.

2.3 Mobile Cloud Services

While pure mobile cloud services are based on data synchronization, there are also other cloud services in each layer of the cloud computing domain (IaaS, PaaS, and SaaS) that might enable the mobiles for increasing their functional capabilities. These services mainly offer processing data-intensive tasks that are high demanding for a handset. Cloud services can be accessed as SaaS using mobiles if the vendor provides WAP content for delivering the service via Web browser. Common mobile SaaS includes those services released by vendor such as Facebook, Google, Apple, etc. The development of mobile applications that involves the use of cloud services in the infrastructure level is limited to the set of tools provided for each specific cloud vendor.

Currently, the Google Application Engine and the Apple Cloud are the most prominent cloud providers that enables the consumption of cloud services from the handset, since their cloud solutions are completely integrated for their own mobile platforms (Android and iOS respectively).More detailed description of mobile cloud services in section 4 while describing MCM.

In behalf of keeping open standards in cloud technologies, some open source communities have been developing libraries that enables the communication with multiple clouds. However such libraries are poorly documented and in some cases are not suitable for deploying within the mobile operating system. Among them we can mention the following.

2.3.0.3 Jets3t

jets3t (20) (Java S3 toolkit) is a library developed by James Murty for accessing the simple storage service of Amazon. jets3t can be easily extended for accessing Walrus storage service of Eucalyptus. Latest version of jets3t enables accessing the Google storage service. its provides support connecting S3 using the Android platform.

2.3.0.4 Jclouds

its another open source library that aims the support for accessing multiple clouds (multi-cloud-API). Latest version of jclouds enables connecting with vendor such as Amazon, GoGrid, Azure, vCloud, and Rackspace. jclouds (21) is not compatible with any mobile platform, and thus it can be use only for the development of pure Web applications.

2.3.0.5 Typica

This library is able to access the computational service of Amazon EC2 and Eucalyptus. its recommended by the open source community to use it altogether with jes3t for developing rich applications based on the consumption of cloud services. However, typica API is not providing support for mobile platforms. Hence, such applications may not be achieved from the handset.

2.3.0.6 Amazon Navite API

Amazon provides a set of Native APIs that can be integrated with the Eclipse environment for the development of applications. Such set of tools provides connectivity with services such EC2, EBS, etc. However, among those set of libraries, just the API for accessing S3 services provides integration with the Android platform.

2.3.1 Synchronization in Mobile Cloud Services

Clouds are looking forward to the mobile domain, having their expectations focused in the idea of data synchronization services. Mobile sync refers to the synchronization of data in the handset with a server and a portal in the cloud (figure). This kind of approach takes advantage of the cloud allowing the centralization of data resources (contacts, e-mail, pictures, etc.) in the storage service, from several sources (social networks, email providers, etc.), for being accessible via SyncML (22) (Synchronization Markup Language) protocol, in such a way that empowers the managing of real-time information from the handset. SyncML is a protocol referenced by the OHA for being used as standard to another synchronization data solutions used for specific vendors. Some of the most popular vendors offering synchronization services are Funambol, Mobical.net, rseven.com and Memotoo.com.

Public cloud vendors offer solutions for synchronization based on their own cloud implementations. For Instance, Google sync (Google), MobileMe (Apple), MyPhone (Microsoft) Ovi Sync (Nokia) etc. In some cases, such solutions are inherent for devices such Android and iPhone. Moreover, providers as Apple unable another kind of devices connecting with its synchronization services. Synchronization solutions for private clouds involves the implementation of open source projects, such as Funambol, libsynthesis, Horde, and eGroupWare among others. Funambol is one of the most popular alternatives for synchronization due enables the integration of several existent technologies (SugarCRM, Zimbra, OpenPSA , etc) through the use of connectors that in most cases are developed by open communities.

2.3.1.1 Funambol

Its an open source framework for the synchronization of data between the handset and a private cloud. Funambol enables the synchronization of information, such as

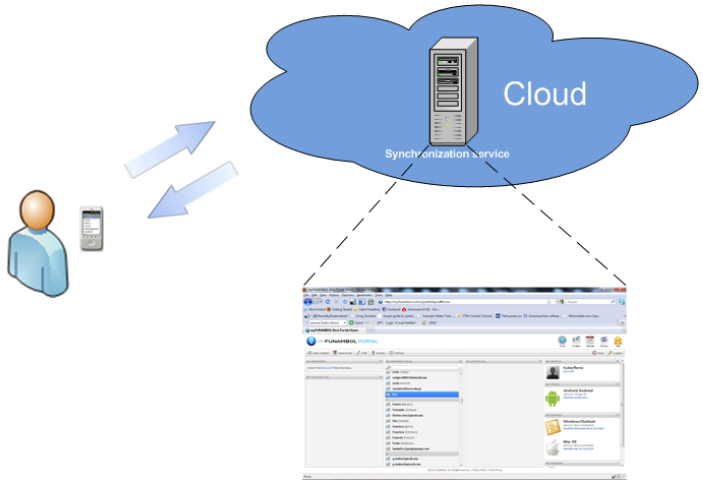

Figure 2.4: Mobile Cloud synchronization architecture

contacts, calendar, images and latest versions include a feature for pushing email to the handset. Recently one study performed by Funambol Inc, situated their solution as the most prominent synchronization technology in the market due several platforms can consume their service without being constrained for an specific cloud implementation (multi-platform client support).

The Funambol core architecture (Server core) consist in three layers, the transport layer, the application layer and the framework layer. The main synchronization functionality relies on the framework layer since contains the synchronization engine altogether with the implementation of SyncML that enables the handset connecting to the sync service. The application layer handles the SyncML messages for being processed. The transportation layer receives messages delivered by protocols, such as HTPP (23), SMTP (24), etc. (General architecture is showed in figure 2.5)

The architecture layer enables Funambol for connecting with other technologies using connectors. Such interfaces allow the pushing of data for synchronization.

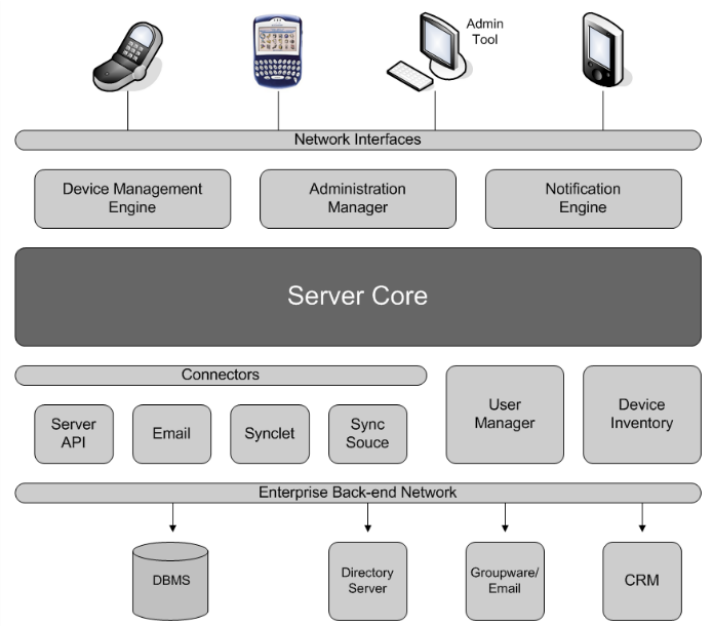

Figure 2.5: Funambol architecture

3

Problem Statement

Since the establishment of SyncML by the Open Handset Alliance (OHA); the synchronization of data is the adopted approach for supplying cloud resources to the handset. Moreover, there is a mobile cloud depency and a set of mobile restrictions that unables the direct invocation of cloud services from the devices. This chapter highlighted some of the main problems for the consumption of cloud services from the mobile phone.

3.1 Research Question

Recently, there has been growing interest in adapting the cloud computing paradigm for delivering services that require high demand. Services provided through the Internet are moving to the cloud due the scalability benefits in infrastructure, the ease in the deploying of services without managing the underlying technology (software and hardware) and the pay-as-you-go model that enables reducing costs. Business companies also are migrating to the cloud for leveraging the potential of their internal infrastructure (using the computing power when is needed). Cloud services may be provided from public cloud vendors (Amazon AWS, Google App Engine, Microsft Azure, etc.) or own private cloud implementations (Eucalyptus). Although, Cloud domain is strongly dominated by proprietary solutions (public clouds). Hence, there exist various cloud architectures that may use different styles (SaaS, PaaS and IaaS) for delivering the cloud supplies. Such architectures are accessible through particular implementations, API set, etc, provided by one specific vendor. Therefore cloud interoperability is not possible and development of applications is tied to a specific cloud provider. For exam-

ple, applications which are developed for consuming services from Amazon must suffer some modifications within the code in order to be compatible with Eucalyptus, even if both shared similar composition.

Same occurs when mobile technologies tries connecting the cloud (set of tools is required). and in some cases mobile platform may be or not supported by the cloud provider. For example, at the time of writing this thesis, Amazon just released a beta support for Android. Moreover, if a mobile applications needs to contact hybrid cloud services; the development becomes a trick process that might not work in some cases, and if works, the result is an ad-hoc heavy application within the handset that is not able to provide a proper interaction with the user.

Since the establishment of SyncML by the Open Hanset Alliance; mobile technologies are taking advantage from the cloud for the provisioning of cloud services based on data synchronization. However, a next generation of mobile technologies is looking towards the cloud for enriching mobile applications functionality utilizing cloud services such Hadoop, EC2, SaaS, S3 etc. for the processing and storing of large amounts of data. Moreover, mobile applications are focusing in the combination of multiple cloud capabilities from multiple clouds for the creation of powerful mashups that are not tied to cloud specifications, and thus can be migrated easily to other cloud solution. Although the direct invocation of those services from the mobile phone is not possible since the handset is holding the request till get a response back from the cloud, This implies that the mobile phone is unable for handle internally both, the managing of multiple cloud requests from one mashup application and the concurrency of several mobile applications that are invoking cloud services.

3.1.1 Mobile Cloud Interoperability

The high penetration in mobiles technologies is fostering the ubiquitous consumption of information from the Internet, and thus services which are offered as SaaS from the cloud are extending their solutions for fitting the mobile demand. This can be contemplated since most of the applications which are available from a Web browser also are available in a mobile version. For example, Twitter, Bebo and MySpace provides WAP content for being consumed from the handset. Similarly, some cloud services are attached to the use of special applications that must be installed within the mobile phone for accessing the cloud resources. As an example, Facebook can be mentioned,

since provides a mobile application that enables the user retrieving information about his or her social updates. Google Calendar is another service which makes use of generic widget applications within the mobile phone for the synchronization of data (activities) with the cloud. However, not all the cloud services are supported for providing mobile version. For example, face.com provides an API for accessing their face recognition service, but such API is not suitable for deploying within a mobile platform.

Cloud services in the infrastructure layer (IaaS) can be accessed using the set of tools (API) provided by the cloud vendor. Those tools enables the applications for storing information in buckets (consist of several objects) and for managing instances from the computational service. However, currently just some libraries provides support for accessing the storage service from the mobile device. Since cloud services usually are consumed by the handset using the synchronization protocol, some of those API are also extending their functionality for fitting such requirement. For Instance, jets3t is a library that is use for accessing Amazon S3 and Eucalyptus Walrus. Moreover it implements a synchronization feature that enables to synchronize files or folders in the buckets with the mobile phone. At the same time, typica is a library that enables accessing Amazon EC2 and Eucalyptus service. It implements routines for retrieving information about the instances that are running at the cloud, Although it doesn't provide support for accessing the computational service from the mobile phone.

While some cloud services can be accessed using one of the approaches described above, there is no way of combining such services in a powerful mashup application. Moreover, those approaches unable the interoperability between clouds due are developed for consuming services from a specific cloud architecture.

3.1.2 Mobile Platform Integration with Cloud Resources

There is a multiple variety of mobile platforms in the market (Symbian, Windows Mobile, Blackberry, etc). However, since the released of iOS and Android, most of the mobile operating systems are decreasing in popularity, and thus the development set of tools for them is diminished or in some cases retired from the development community due the extinction and replacement of the mobile platform. (Symbian case). Most of the cloud vendors provides support for their own mobile OS or for some specific third-party solutions. Moreover, some cloud vendors used to offer compatibility with their cloud services using certain devices and in further versions of the same services

that compatibility was removed or avoided. For example, iTunes store (Apple service) a service for purchasing music online; it used to offer synchronization of the songs with Nokia and Palm. Although, current version of iTunes only allows synchronizing the streaming with iPod or iPad.

Mobile platforms that are supported by the cloud vendors are limited to consuming just certain cloud services, Therefore it is not possible to take the complete advantage of all the capabilities of the cloud . For example, jets3t only allows to access Amazon S3 from the handset. On the other hand, open source libraries which were developed initially for accessing cloud services from standalone applications might be adapted for mobile phones. However, some important problems of adapting those set of tools are concerning to the platform restrictions. Most of the APIs which are available to interact with the public and private clouds are not suitable for directly deploying them into a mobile phone, due to the integration issues with the compiler or other libraries which are required by the cloud API, but are not compatible for mobiles. For example in jclouds API when some dependencies are included within Android, various runtime issues emerge which are not supported by the platform compiler, thus the application gets unable to execute. Moreover, they tend to get outdated and newer versions cannot be deployed on older applications, due to the deprecated methods.

Clearly, mobile platforms are playing a vital role in adopting the cloud as solution for mobile technologies issues concerning storage and processing power. However, the intention of locking cloud services to specific platforms unable innovation of applications in mobile technologies.

3.1.3 Cloud Services Invocation from the Mobile Device

Mobile technologies are accessing services from the cloud using a synchronization approach. Such strategy consist on keeping data consistency between the handset and the cloud; the data can be manage using a portal located at the cloud. However, an innovative type of the mobile applications are trying of using the cloud resources for performing data-intensive processing that may enrich mobile applications with more functionality. Context-aware applications are those that might benefit more from such processing due they collected large amount of data through sensors (accelerometer, magnetic field, etc) that needs to be processed for learning from the user context.

Cloud services such Hadoop is considered for delivering context-aware services from the cloud.

However, analyzing large amount of information is time consuming, and thus the invocation of a cloud service from a handset is not proper due the waste of energy required in the request offloading and the waiting for a response back. Moreover, such waiting time is not tolerable for the user perspective and mobile application usability. Therefore, a mobile cloud invocation must be handle by the mobile as any other local application. Another problem concerning the invocation of cloud services from the mobile is related with concurrency of applications. Since a simple direct invocation requires an immediate response for releasing the handset, just one task can be performed at the time.

3.1.4 Summary

To counter the problems with the interoperability across multiple clouds, to perform data-intensive processing invocation from the handset and to introduce the platform independence feature for the mobile cloud applications, the following thesis discusses a Mobile Cloud Middleware (MCM). The middleware provides a unique interface for mobile connection and multiple internal interfaces and adapters, which manage the connection and communication between different clouds. The MCM capabilities for managing the resource intensive tasks can easily be envisioned in several scenarios which are discussed in further sections as case of study.

4

Mobile Cloud Middleware

Several are the issues that were discussed in previous chapter, regarding the use of cloud services from the mobile. Most of the cloud solutions offered for handsets consist in applications that make use of synchronization of data. However, mobile technologies are drawing the attention to the cloud mainly for the processing and the storing of large amount of data using services such Hadoop (25), EC2, EBS, etc. Moreover the combination of those services with the existent SaaS solutions foster the development of rich mobile applications. This chapter introduces the Mobile Cloud Middleware (MCM) for solving most of the issues described above.

4.1 A Generic Middleware Framework for Mobile Cloud Services

Mobile cloud services use the shared pool of computing resources provided by the clouds to get the process and storage intensive tasks done from smart phones. Generally mobile applications focus at enriching their functionality to a mashup application, using the cloud services. However, clouds are looking forward to the mobile domain, having their expectations focused in the idea of data synchronization services. Mobile sync refers to the synchronization of data in the handset with a server and a portal in the cloud. This kind of approach takes advantage of the cloud allowing the centralization of data resources (contacts, e-mail, pictures, etc.) in the storage service, from several sources (social networks, email providers, etc.), for being accessible via SyncML protocol, in such a way that empowers the managing of real-time information from the handset.

Some of the most popular vendors offering synchronization services are Funambol, Mobical.net, rseven.com and Memotoo.com.

While pure mobile cloud services are based on data synchronization, there are also other cloud services in each layer of the cloud computing domain (IaaS, PaaS, and SaaS) that might enable the mobiles for increasing their functional capabilities. These services mainly offer processing data-intensive tasks that are high demanding for a handset. This can be observed as most of the SaaS are extended to fit the mobile demand. For example, both Google docs and Zoho provide a mobile version of their office suites, also Picasa and flickr offer a service to visualize pictures that fit the mobile screen. Several other cloud services are accessible through special applications, specific to different mobile platforms, which can be bought in the mobile application market, for example, Google Analytics service can retrieve reports from applications such as GAnalyticz for Android, I Spy Analytics for Iphone and Google Analytics Mobile for Windows Mobile.

Mobile applications can also access basic infrastructure cloud services (IaaS) using APIs, which are provided in some cases by the cloud vendors or by one open source community. Most of the APIs, which enable interacting with the public and private clouds, are not suitable for directly deploying them into a mobile phone. The reason being; the integration issues with the compiler or other libraries which are required by the cloud API, but are not compatible for mobiles. For example in jclouds API when some dependencies are included within Android, various runtime issues emerge which are not supported by the platform compiler, thus the application gets unable to be executed. There exist alternative solutions, which can solve the problem of interacting with the cloud from the mobiles. APIs such as jets3t and typica can be deployed on Android. However, it is not possible to take the complete advantage of all the capabilities of the cloud, in such cases, as open source APIs, in the mobile environment, have certain limitations and some functionality is lost in the process. Moreover, cloud vendors are generally observed to be slow in providing APIs for multiple mobile platforms. For example, at the time of writing this thesis, Amazon just released the mobile API for Android and still in beta.

While several mobile cloud services are existing today, most of them are bounded by numerous constraints like the mobile platform restrictions, cloud provider's technology choices etc. They provide proprietary APIs and routines to consume the services.

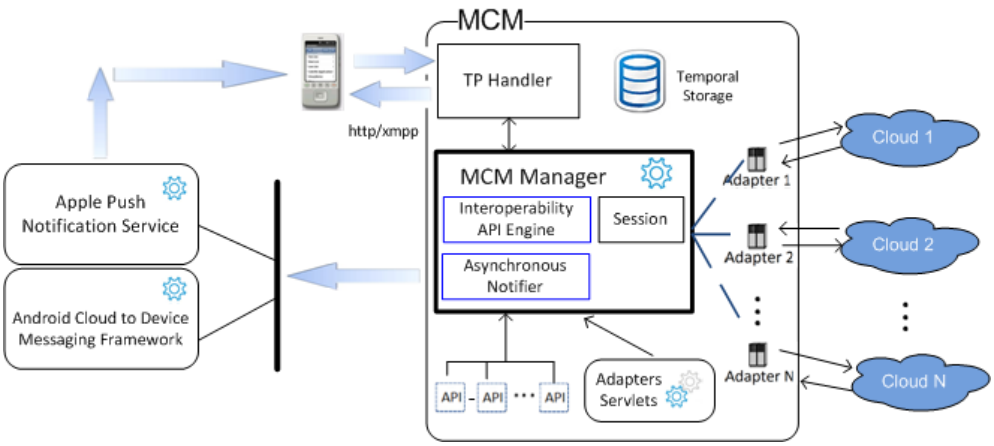

Figure 4.1: Architecture of the Mobile Cloud Middleware

Therefore, cloud interoperability is not possible and when a lighter mobile application is to be created, it has to be developed for a specific cloud provider. For example: an application created on Android using jets3t to access S3 from Amazon, and Walrus from Eucalyptus, has to be suffer some changes in its configuration when it tries to access each of the cloud storage services. Even though Eucalyptus is compatible with Amazon infrastructure, there is no full integration between them. To address most of these problems, the Mobile Cloud Middleware (MCM) has been designed.

4.1.1 MCM Architecture and Realization

MCM is introduced as an intermediary between the mobile phones and the cloud. The architecture is shown in figure 4.1. MCM fosters mobile platform heterogeneity and the combination of different cloud services into a mobile mashup application. When an application tries to connect to a basic cloud service, it connects to the TP Handler component of the middleware, which receives the request. The transportation handler can receive the requests based on several protocols like the Hypertext Transfer Protocol (HTTP) or the Extensible Messaging and Presence Protocol (XMPP) (26). The request is then processed by the Interoperability API engine, which selects the suitable cloud API and creates a unique adapter that ensures the transactional process with the cloud.

When the request is forwarded to the MCM Manager, it first creates a session assigning a unique identifier for saving the system configuration of the handset (OS, clouds credentials, etc.) and the service configuration requested (list of services, cloud

providers, types of transactions, etc.) in a temporal storage space, respectively. The identifier is used for handling different requests from multiple mobile devices and for sending the notification back when the process running in the cloud is finished. Later, the interoperability API engine verifies the service configuration for selecting the suitable API, depending on the cloud vendor. A temporal transaction space is created for exchanging data between the clouds. The aim of the temporal space is to avoid offloading the same information from the mobile, again and again.

Once the interoperability API engine decides which API set it is going to use, the MCM Manager requests for the specific routines from the Adapter Servlets. The servlets contain the set of functions for the consumption of the cloud services. Finally, MCM Manager encapsulates the API and the routine in an adapter for performing the transactions and accessing the SaaS. The result of each cloud transaction is sent back to the handset in a JSON (JavaScript Object Notation) (27) format, based on the application design.

Since the completion of a cloud task, most often, is time consuming, it is not logical to make the mobile application waiting for the response and even not tolerable from the user's perspective. To support this, MCM supports asynchronous mobile cloud service invocation for both Android and iOS platforms. The asynchronous services thus enable to perform concurrent activities at the device. In this process, when the mobile terminal sends a request to the middleware, the mobile immediately gets a response that the transaction is in progress. Now the mobile device can continue with other activities. When the transaction is finished at the MCM, the middleware has to follow a specific protocol to notify the mobile terminal about the updates.

MCM is implemented in Java as a portable module based on Servlets technology, which can easily be deployed on a Tomcat Server. Mobile cloud services using APIs from Amazon EC2, S3, Eucalyptus, Google and some open source cloud projects like Eucalyptus are considered. From the cloud infrastructure level, jets3t is the API within MCM that enables the access to the storage service from Amazon and Google. jets3t is an open source API that handles the maintenance for buckets and objects (creation, deletion, modification). A modified version of the API was implemented for handling the storage service of Walrus (Eucalyptus storage service). Latest version of jets3t handles synchronization of objects and folders from the cloud. typica API and the Amazon API are implemented to manage (turn on/off, attached volumes) the instances

from Eucalyptus and EC2 respectively. MCM also has support for SaaS from Facebook, Google Analytics and Picasa.

4.1.2 Asynchronicity for Handling Process Intensive Mobile Cloud Services

Some mobile applications, whose functionality depends on mobile cloud services, are able to provide a tolerant response time to the user. For example, searching for a location in Google Maps or requesting for a preview of a picture in flickr. However, when a mobile application needs to perform a task which is expected to be time consuming, it cannot hang the mobile phone until the response arrives. Mobile devices tend to get stuck if the computation offloading requires long waiting, and in some cases unable the complete OS or just don't allow performing another task at the same time.

As a solution to address these issues, MCM implements an asynchronous notification feature for mobile devices that foster the de-coupling between the client and server. A mobile application can delegate the execution of a remote task in the local background to be processed in the cloud. Once the process is finished at the cloud, a notification about the result of the task is sent back to the handset. The approach can also be used within a mobile device to concurrently execute several tasks in multiple clouds.

However, sending the notification back to the mobile device is a tricky process. To have a generalized solution, one can make the mobile check for the status of the service regularly. However, this puts a lot of load on the mobile networks. Alternatively, we can make the mobile device a service provider, which was studied in mobile web service provisioning project. The study developed a Mobile Host, which can be used in providing web services from the smart phones. When the smart phone acts as a server, the mobile cloud service response can be sent directly to the device. However, the current implementations of the Mobile Host are available only for PersonalJava and J2ME platforms. Considering this, we developed solutions to provide the asynchronous mobile cloud service invocation, specific to Android and iOS platforms, due to their popularity in the mobile market, using AC2DM (28) (Android Cloud to Device Messaging Framework) and APNS (29) (Apple Push Notification Service), respectively.

4.1.2.1 MCM and AC2DM

Android Cloud to Device Messaging Framework is a service that enables to send data from servers to applications on Android devices. AC2DM is a lightweight mechanism which enables the servers to communicate asynchronously with mobile applications running on Android OS. This communication is established between third-party servers to send lightweight messages to control the behavior of their Android applications or just to notify about the result of any process which was running on the server and the application needs to be notified about. The application does not need to be running to handle the incoming messages since they are handled by the system through an Intent Broadcast. Once the message is receive the system will wake up the application via Intent Broadcast passing the raw message data received straight to the application.

MCM implements the communication with the third party servers to inform about the completion of the tasks in the cloud. In order to receive messages from the third party servers to the Android mobile the mobile must be registered against the C2DM Google services which will give one Registration ID to the device. Once the device gets the Registration ID it must send it to the Third-party server, as the server uses it since it to route the messages by the C2DM Google Services. This Registration ID is updated periodically and the third-party server needs to be notified about it. In order to send a message to a particular device, the third-party server sends one Https request to C2DM Google services sending the Device Registration ID, the Authorization token associated to the C2DM service, the message and a delay while idle time. The C2DM Google services will send the message to the mobile application which will be handled by the system through an Intent Broadcast and will perform the actions associated to the message.

4.1.2.2 MCM and APNS

Applications on Apple mobile devices running iOS 3.0 or newer can receive asynchronous messages provided through Apple Push Notification Service (APNS). APNS messages are sent through binary interface (gateway.push.apple.com:2195 and gateway.sandbox.push.apple.com:2195) that uses streaming TCP socket design. Forwarding messages to device happens through constantly open IP connection. TLS (or SSL) certificate (provisioned through iOS developer portal) is needed for creating secure

communication channel and for establishing trusted provider identity. To avoid being considered a DoS attacker, using one connection for multiple notifications, rather than creating new for each one, is desired.

APNS has a feedback service that records failed notification delivery attempts. This information should be checked regularly to avoid sending messages to devices that do not have the targeted application installed anymore. For the same reason application should register itself at MCM for notifications at each start by providing at least it's device token that it received from APNS. Device token is 32 byte hexadecimal number that is unique for an application on a device and does not change.

Messages sent to iOS devices via APNS consist of JSON payload with maximum length of 256 bytes. Within message payload, values for keys alert, sound and badge can be used to customize the message alert being shown to user upon receiving it. Because the payload size is limited, it is used to provide enough information for the application to make request for additional details. When the device is unable to receive notifications for some time (e.g. due to being offline or switched off) and multiple notifications have been sent, only the last one is guaranteed to be delivered.

4.2 MCM and Funambol

Funambol is an open source framework for the synchronization of data between the handset and a private cloud. Funambol enables the synchronization of information, such as contacts, calendar, activities, images and latest versions include a feature for pushing email to the handset. Funambol can be easily extended for the integration with other technologies since implements connectors for pushing data into the Funambol core architecture. Open source solution provided by Funambol Inc, consist in the ds-server, the Web demo portal and the server management console.

The ds-server is executed using Tomcat and the Web demo portal is based on Servlets technology. Therefore, the integration with MCM consist in pushing information about the cloud tasks monitored by MCM into an interface that enables the synchronization of activities with the handset. Hence, the status of each cloud task is delivered as activities that were accomplished by the user. Since the integration of both solutions needs refinements and more development, this feature is introduced as proof of integration concept.

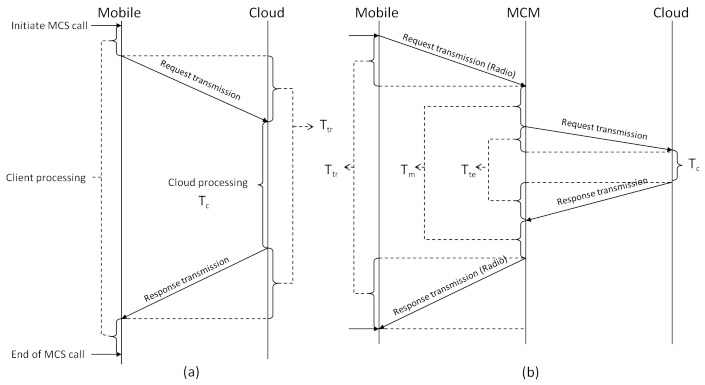

Figure 4.2: Mobile cloud service invocation cycle: Activities and timestamps in (a) Regular case (b) With the MCM in place

4.3 Analysis of MCM

Once the MCM prototype was designed and developed, it was evaluated to verify its performance. The performance model and the analysis are addressed in the following subsections.

4.3.1 Performance Model of the MCM

In the regular mobile cloud service invocation cycle, as shown in figure 4.2a, the total time of invocation includes the transmission delays and the time to process the service at the cloud. So the time taken to handle a mobile cloud service (MCS), T_{mcs_r} can be calculated as:

$$T_{mcs_r} \cong T_{tr} + T_c \tag{4.1}$$

Where, T_{tr} is the transmission time taken across the radio link for the MCS invocation. The value includes the time taken to transmit the request to the cloud and the time taken to send the response back to the mobile. Apart from these values, several parameters also affect the transmission delays like the TCP packet loss, TCP acknowledgements, TCP congestion control etc. So a true estimate of the transmission delays is not always possible. Alternatively, one can take the values several times and can consider the mean values for the analysis. T_c is the time taken to process the actual

service at the cloud. $\cong$ is considered in the equations as there are also other timestamps involved, like the client processing at the mobile phone, as shown in figure 5.7a. However, these values will be quite small and cannot be calculated exactly.

When the MCM is introduced to the invocation cycle, the invocation cycle transforms to the one shown in figure 5.7b. Here, the total mobile cloud service invocation time, T_{mcs_m} is:

$$T_{mcs_m} \cong T_{tr} + T_m + T_{te} + T_c \tag{4.2}$$

T_{tr} becomes the transmission time across the radio link for the invocation between the mobile phone and the MCM. T_m is the time taken to process the request at the middleware. T_{te} is the transmission time across the Internet/Ethernet for the invocation between the middleware and the cloud. T_c stays the same as in equation 4.1.

From equation 5.1, we can observe that extra delays are included to the invocation cycle with the middleware in place. However, apart from the main benefit of interoperability that is brought to MCS, several observations also make the middleware a better solution even in terms of the performance. Consider the case of synchronizing the data across multiple clouds or having to perform parts of the service across multiple clouds; In this case, the regular mobile cloud service invocation cycle time becomes,

$$T_{mcs_r} \cong \sum_{i=1}^{n} (T_{tr_i} + T_{c_i}) \tag{4.3}$$

Where as in the case where the MCM is involved in the invocation cycle,

$$T_{mcs_m} \cong T_{tr} + T_m + \sum_{i=1}^{n} (T_{te_i} + T_{c_i}) \tag{4.4}$$

However, the transmission delays across the radio link are far greater than the transmission delays across the Internet due to the still lesser transmission capabilities across the mobile networks. Assuming, $T_{te} \lll T_{tr}$, the middleware solution quickly outperforms the regular solution as the value of i increases in equations 4.3 & 4.4.

While performance model of the above synchronous cases is fine, the asynchronous mobile cloud service invocation, addressed in section III.A, works the best for the mobile terminal. Here the mobile phone just sends the request and gets the acknowledgement back. The response is followed later. MCM will be at its conceptual best with this

case, as the middleware takes care of the process completely, freeing the mobile terminal. However, in this asynchronous invocation, coming up with proper performance model is tricky. The delays completely depend on external sources like the latencies with AC2DM Framework and the respective clouds. The phone is rather free to continue with its tasks, so not much load on it. In simple, in this case T_{tr} increases as it has 2 invocations in radio link, and T_m increase as MCM does more processing. And both T_{mcs_r} and T_{mcs_m} rather stay constant for however big the mobile cloud service may be.

4.3.2 Performance Analysis of the MCM

To analyze the performance of the MCM, the cloud data synchronization application explained in section III.B is considered. Several photos were taken with the mobile phone, of multiple resolutions, thus varying the size of the image. Figure 4.3 explains the scenario in detail. HTC desire phone (30), with a 5 megapixel color camera with auto focus and flash was considered for the analysis. It has a CPU speed of 1GHz, 576 MB of RAM and storage that can be extended up to 32GB. The application was developed based on the Android platform, compatible with Android 2.2 API. Wifi connection was use to connect the mobile to the middleware or the cloud. So, test cases were taken in a network with an upload rate of $\approx$ 1409 kbps and download rate of $\approx$ 3692 kbps, respectively. However, as mentioned already, estimating the true values of transmission capabilities achieved at a particular instance of time is not trivial. To counter the problem, we have taken the time stamps several times (10 times), across different parts of the day and the mean values are considered for the analysis.

As part of the application, the photos taken by the mobile phone were synchronized across multiple clouds. The extra latencies added by the MCM to the invocation cycle are calculated. Figures 4.4 and 4.5 show the times taken to upload the photos to Amazon EC2 and the SciCloud, respectively. SciCloud is a private cloud at University of Tartu, built based on the Eucalyptus technology. Thus the considered service becomes a hybrid cloud service, as it uses both the public and private clouds.

From figures 4.4 and 4.5, we can observe that, even though a small latency is added to the invocation cycle with the middleware, the total time of invocations vary uniformly with the ones of regular invocation cycle. The phenomenon is observed across both the public cloud and the private cloud. From this analysis we could observe that the MCM shows reasonable performance levels, thus validating the proof of concept.

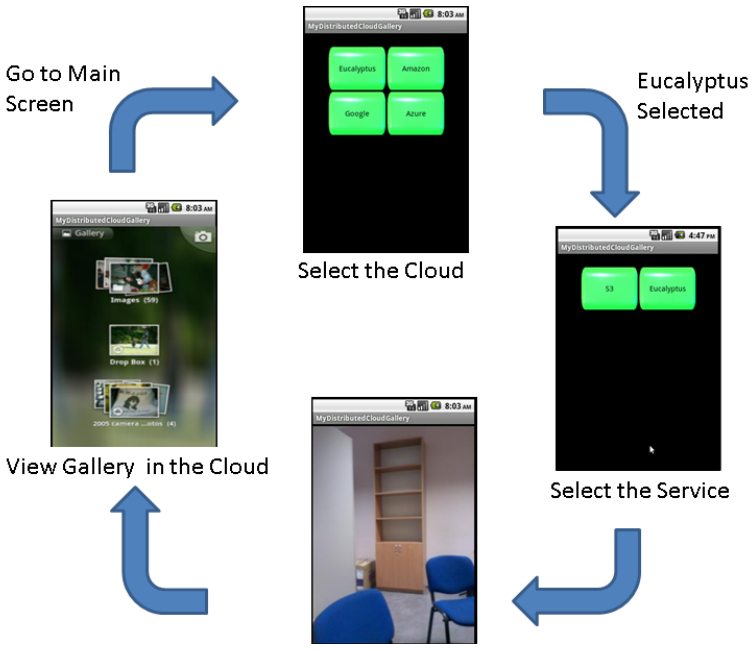

Figure 4.3: Test case scenario used for the performance analysis of the MCM

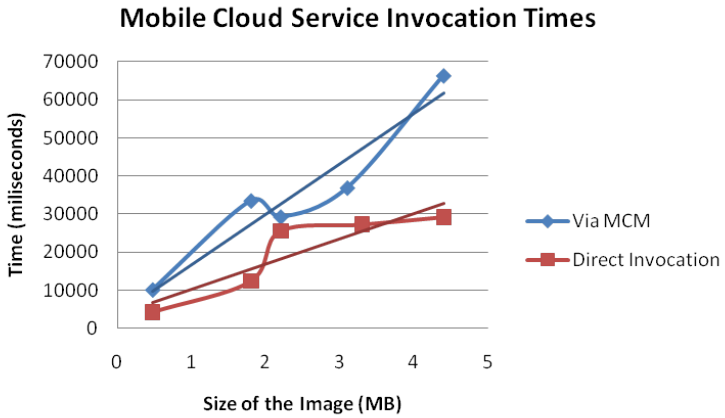

Figure 4.4: Mobile cloud service invocation times for the test case scenario in public cloud (Amazon)

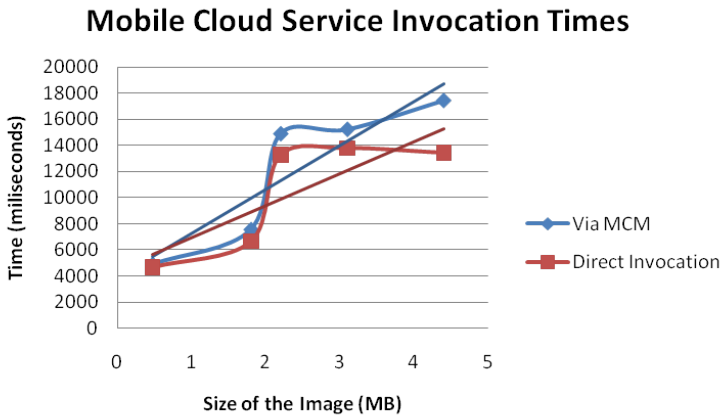

Figure 4.5: Mobile cloud service invocation times for the test case scenario in private cloud (SciCloud)

All the experiments addressed here are conducted 10 times and the mean values are considered for the analysis, to have statistically valid results.

4.4 Related Work

Even though cloud computing domain is getting popular, still most of the applications are dependent on the proprietary API of the cloud providers. Several open source projects like jets3t, jclouds etc have been working to merge the functionality of different cloud APIs in order to create a single API which can be used as a generic one for accessing any cloud. jets3t is one of the projects, which has produced an open source API written in Java which allows the applications to use S3 from Amazon and the storage service, Walrus from Eucalyptus. Latest version of jets3t also includes support for the Google storage service (Google for developers).

Jclouds is a multicloud API which aims to combine all the different routines for connecting and consuming cloud services using just one library. Jclouds gathers all the common functionality in core libraries, although special dependencies have to be included depending on which cloud the application is supposed to access. Similarly, typica is a Java client library which can be used for consuming variety of cloud services from Amazon Services like the EC2 and S3. Other projects working with similar targets are Deltacloud (31) and Dasein (32).

However, not all of these projects have implementations compatible with mobile platforms like the Android. Only, jets3t can be used in the development of mobile applications for Android OS. Some parts of the other open source APIs can be used for the development of mobile applications but most of them are not suitable at all, due to the dependency issues which are not supported within the mobile OS. So the research with mobile cloud computing domain is still in its infancy. As far as developing community based applications for mobile devices is concerned, produced a report of feasible mobile cloud computing based application domains.

Another project which is looking to use the power of the cloud to enhance mobile experience is the Zeus (33) project, which is targeting the use of virtual machines from the mobile phones and proposing a new concept known as VMaaS (Virtual Machine as a Service).

Middleware approaches similar to MCM have also been addressed in the literature. MCCM (Mobile Cloud Computing Middleware) (34) is a project which involves the use of a middleware, standing between the mobile and the cloud, for the consumption of web services (WS) in a mobile mashup WS application. It handles creating user profiles from the context of the mobile phone, storing the system configuration (pre-defined set of WS which can be consumed from the handset) and the service configuration (information needed for constructing a request and combining services) respectively, for managing existing resources in the Internet Cloud. However, such middleware solution and the API support seems to be tightly coupled.

Cloud agency (35) is another solution that aims to integrate GRID, cloud computing and mobile agents. The specific role of GRID is to offer a common and secure infrastructure for managing the virtual cluster of the cloud through the use of mobile agents. Agents introduce features that provide the users a simple way for configuring virtual clusters. Agora (36) is another middleware solution which is in the development, which will enable new large-scale mobile-cluster applications and will use mobile devices as nodes of a large-scale cloud-computing infrastructure. Agora will enable the devices to work together seamlessly. However, a concrete implementations cannot be found yet.

However, MCM is much more scalable and portable solution which manages all the transactions with multiple clouds in a transparent way. It allows to build more lighter applications for mobile devices, as the developers do not have to deal with several APIs,

it just have to implement the functions provided as a web service at the middleware. MCM mainly enables interoperability across multiple cloud architectures. One feature which really separates it from other approaches is its support for asynchronous push notification, which frees the mobile resources during most of the invocation process. Moreover, our earlier research also involved working with middleware for mobile web services, where Srirama et al. have realized a mobile web service mediation framework (MWSMF) that helps in offering proper quality of service (QoS) and discovery mechanisms for the services being provided from the smart phones. MWSMF is shown to be reasonably scalable and our future research will try to add the MCM as a component to the MWSMF.

4.5 Summary

While the prototype is working properly with the traditional web technologies like the HTTP and servlets, the future research will be focus on extend the architecture to better suite the cellular network, by providing the access to the MCM via the XMPP protocol. XMPP is an open technology for real-time communication, which powers a wide range of applications including instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware and generalized routing of XML data. However, this requires the protocol to be implemented for the mobile platforms we have considered in the analysis; Android and iOS.

Since scalability is the main concern for centralized middleware, and the current Java implementation cannot easily scale, due the language limitations. Another improvement involves the re-implemention of the middleware using Erlang since it is more suitable for highly concurrency programs.

5

Case Studies

The development of mobile applications that requires data-intensive processing and at the same time that keeps a tolerable interaction with the user, its feasible though the use of MCM. This chapter describes two Android applications. Zompopo a mobile application that foster the provisioning of context-aware functions from the cloud and CroudSTag that enables the face recognition process from the handset for the creation of social groups in Facebook.

5.1 Zompopo: Mobile Calendar Prediction based on Human Activities Recognition using the Accelerometer and Cloud Services

Lately, mobile devices have become an indispensable tools in everyday life due enables the ubiquity access for storing user's information (contacts, agenda etc.) and for executing low demanding computational tasks (calendar, text editor etc.). Moreover, handsets are too attached to the user that may be used for capturing his or her context for enhancing the mobile applications with proactive behavior. For instance, the light sensor within the smartphones increases or decreases the brightness in the screen depending in the environmental changes with the purpose of saving energy. Services which are used often such a mobile calendar can enrich monitoring the way in which the carrier behaves.

A calendar service for mobiles can be provided locally within the handset as widget application (e.g. Android calendar etc) or externally as mobile cloud service using

SyncML (e.g. Google Calendar, Funambol etc), the difference between them relies in the fact that using an external source multiple handsets can be synchronized with the cloud, meanwhile the local source is only useful for one individual user. Zompopo makes use of the second approach since it uses cloud services (Hadoop) for processing the data gathered by the accelerometer. Zompopo is an application that tries to extend the capabilities of a generic calendar adding a feature that makes use of the accelerometer sensor for predicting the activities which the user will perform during the day based on the sensing of previous week's activities. Since the accuracy of the prediction depends on a set of data collected in advance, the use of the Zompopo application is restricted to collecting information one week before the activation of the prediction feature. The following description assumes that the information was already collected.

While Zompopo is executing in the handset background; the data from the accelerometer is gathered and stored in a SQLite database (the accelerometer analysis is discussed in detail in section 3). By default the accelerometer is always sensing environmental changes that is appended to the database file and then offloaded from the mobile to the cloud storage once per day (23:00 pm). The file is uploaded though MCM (Mobile Cloud Middleware) to the cloud with an unique Id that consist in the date plus the prefix zompopo. For example: The offloading of today was stored as "09-05-2011-zompopo". At the beginning of each day (generally 7:00 am), Zompopo sends a request to MCM for obtaining the set of activities to be included in the calendar. Since MCM implements an asynchronous notification feature for decoupling the handset from the mobile cloud services; the hadoop task for analyzing all the set of historical files is delegated to MCM, releasing the mobile from the activity. The progress of the task is monitored by MCM, which informs the user through a notification message when the task is finished along with the information about its final result (refer to figure 5.1). MCM is discussed in detail in section 4.

Once the handset is notified about the results, Zompopo shows a screen with the list of suggested activities (hour + name of the activity) that could be included in the daily calendar. Since Zompopo was developed for Android; the activities are created using the default calendar application which comes with the OS. The android calendar allows to use SyncML for the synchronization with Google calendar service, and thus changes are replicated to the cloud calendar service automatically. However, the creation of

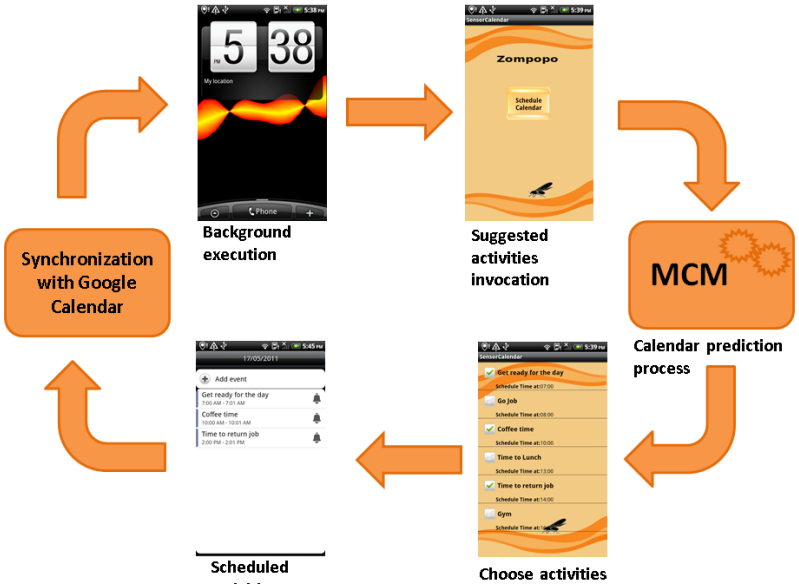

Figure 5.1: Zompopo application flow

activities is also possible from MCM as is able to access Google cloud services; therefore Zompopo is not tied to the mobile platform and can be easily extended.

5.1.1 Human Activities Recognition Using the Accelerometer and Hadoop Processing Service

Nowadays mobile devices are equipped with a variety of sensors (GPS, magnetic field, etc) that enrich the mobile applications with location awareness and sensing capabilities. Those advances enable fitting contextual requirements for improving the quality of service in the applications as it allows adapting the interaction between the handset and the user in real-time. For example, a sensor such as the accelerometer is used for sensing how the user is holding the handset, and thus changing the position of the screen as a result. The accelerometer is a sensing element that measures the acceleration associated with the positioning of a weight in which it has being embedded (device). Depending on the number of axes, it can gather the acceleration information from multiple directions. Generally a triaxial accelerometer is the most common in mobiles from vendors such as HTC, Samsung, Nokia etc. Therefore, acceleration can be sensed on three axes, forward/backward, left/right and up/down. For example: in

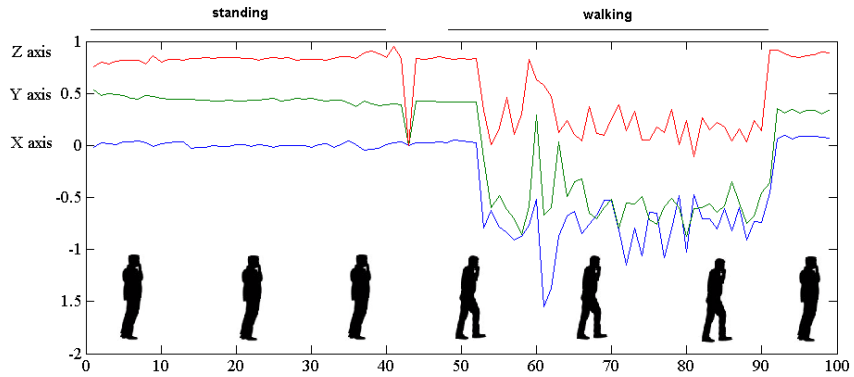

Figure 5.2: 3-axis readings for different activities

the case of a runner, up/down is measure as the crouching when is warming up before starting to run, forward/backward is related with speeding up and slowing down, and left/right involves making turns while he is running.

While the accelerometer is able to track information for the recognition of multiple human activities (walking, running, etc) for an immediate response, each activity is differentiated according to the behavior of the data collected by the accelerometer as shown in figure 5.2. In the context of Zompopo it can be used altogether with cloud services for the prediction of repetitive activities based on a historical set of data. The accelerometer provides information across time related with acceleration along x axis, acceleration along y axis and acceleration along z axis. This information may be used for the identification and classification of certain patterns which are defined in the Zompopo criteria as; standing, walking and running. However, for performing such analysis a classification algorithm using Hadoop is used, which is shown in figure 5.3. Hadoop is a framework that provides support for the analysis of data-intensive distributed applications (thousands of nodes) within the cloud. The algorithm applies map/reduce for matching the patterns described above, but since the actual aim of the Zompopo is the invocation of data-intensive processing services from the cloud, a basic algorithm is introduced. This algorithm was implemented using Hadoop 0.20 and Java as programming language.

The MapReduce algorithm consists of two basic steps, the Map and the Reduce functions. The Map function takes one set of key value pairs and maps them to intermediate key pairs. The intermediate key pairs are sorted and grouped together by the

```
Classification Algorithm(csvFile, inputFile, outputFile, threshold)

1. Map from csvFile to SequenceFile
     writeNumber(csvFile, sequentialFile)
          writer.append(time, num);

2. Generate intermediate keys from Sequence File.
     map(LongWritable time, DoubleWritable x, Context context)
          context.write(new_key, new_value);

3. Reduce - calculate statistical measures and infer the move
   actions.
     reduce(LongWritable key, Iterable<DoubleWritable> values,
          Context context)

   4. Calculate Mean values
          mean = calulcateMeanValues(values)

   5. Calculate Standard Deviation
          sd = calulcateStandardDeviation(values,mean)

   6. Determine the value of Action
          if (sd> threshold)
             output[ACTION] = MOVE_ACTIVITY;
          else
             output[ACTION] = NOT_MOVE_ACTIVITY;

   7. Generate final keys
          context.write(key, new ArrayWritable(output));
End of the algorithm
```

Figure 5.3: Classification algorithm using Hadoop map/reduce

```
writeNumber(csvFile, sequentialFile)
  createSequentialFile()
  foreach line in csvFile:
      sequentialFile.append(time,[x,y,z])
  done
end
```

Figure 5.4: Sequential file procedure

```
map(time, [x,y,z])
  emit Intermediate(time, x);
end
```

Figure 5.5: Segregation of accelerometer data based on x axis

framework and passed to the reduce function. The Reduce function takes the intermediate key pairs and produces the output. The input process is showed in figure 5.4 and uses a CSV file to start the algorithm. Each line within the file contains the following information <index, time, x, y, z>, where time is measured in hours, x, y and z are the 3 axis measured by the accelerometer. Those data is mapped individually to one key value with the following structure <time, [x, y, z] >to produce one Sequential File that is the input for the MapReduce process.

The Map function takes each key <time, [x, y, z]>from the Sequential File and creates one temporary key <time, x>(figure 5.5). The value of x is considered more representative than y and z since x measures the change of position when person is moving forward or backward and the prediction is based on the idea of movements that involves the carrier locomotion from one place to another. Thus, the recognition is based on x axis. In future improvements of the algorithm the values y and z will be considered to produce more accurate results.

Later, the Reduce function receives the temporary keys grouped and sorted by time. Each key represents one set of x values and each key is processed by one Reducer. Two statistical measures, the mean and the standard deviation are used for analyzing the data, and thus determining whether the user is moving or not. The standard deviation indicates how the data is spread out over a range of values. Based on figure 5.2 the more spread out the values are the more the user moves and in the opposite way the more the values are close to each other the less the user is moving. The Reduce function

```
reduce(time, xValues, threshold)
  meanValue = calculateMean(xValues)
  sd = calculateStandardDeviation(xValues)
  if (sd > threshold)
    emitFinal(time,mean,standardDeviation,"Moving")
  else
    emitFinal(time,mean,standardDeviation
              ,"Not Moving")
end
```

Figure 5.6: Reduce function

(figure 5.6) calculates the two statistical measures and uses the standard deviation to determine if the user is moving or not for the given set of values. One threshold value for the standard deviation is defined with a value of 1 for this experiment. If the standard deviation is below the threshold values the algorithm infers that the user was not moving. If the standard deviation is greater than the threshold value it means the data is spread enough to infer that the user was moving by the time the data was measured. The Reduce produces the output in CSV file with information such <time during the day, Accelerometer Measure, Standard Deviation, Action>, where accelerometer Measure is the mean value of the x values received by the Reducer and Action is the activity inferred by the algorithm.

5.1.2 Zompopo Performance Model: Asynchronous Service Invocation

On the basis of the functional Zompopo prototype, the application was tested extensively for understanding its interaction performance with the user. The performance model and the analysis are addressed here. Figure 5.7 shows the sequence of activities that are performed during the execution of the application. Here the total application duration i.e. the total mobile cloud service invocation time, is:

$$T_{mcs} \cong T_{tr} + T_m + \Delta T_m + \sum_{i=1}^{n}(T_{te_i} + T_{c_i}) + T_{pn} + T_{sync} \tag{5.1}$$

Where, T_{tr} is the transmission time taken across the radio link for the invocation between the mobile phone and the MCM. The value includes the time taken to transmit the request to the cloud and the time taken to send the response back to the mobile. Apart from these values, several parameters also affect the transmission delays like the TCP packet loss, TCP acknowledgements, TCP congestion control etc. So a true

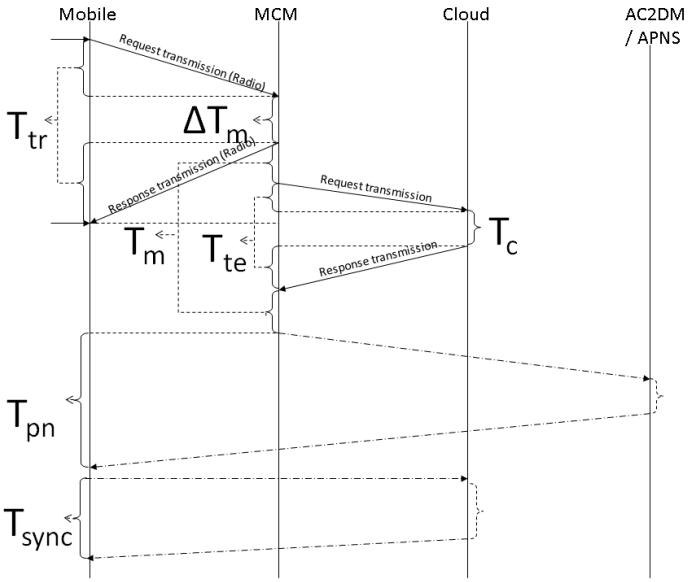

Figure 5.7: Mobile cloud service invocation cycle: Activities and timestamps

estimate of the transmission delays is not always possible. Alternatively, one can take the values several times and can consider the mean values for the analysis. T_m is the time taken to process the request at the middleware. ΔT_m is the minute extra latency added to the performance of the MCM, as the mobile is immediately notified with the acknowledgement. T_{te} is the transmission time across the Internet/Ethernet for the invocation between the middleware and the cloud. T_c is the time taken to process the actual service at the cloud. This process is repeated several times in the Zompopo application, as it is contacting different clouds like Eucalyptus, Google and Amazon. Hence the sigma is considered in the equation.

Similarly, T_{pn} represents the push notification time, which is the time taken to send the response of the mobile cloud service to the device via the AC2DM. Once the notification is received by the mobile phone the activities are created locally in a generic calendar. Since the information calendar is an inherent Mobile sync feature for Android; an extra time is introduced, T_{sync} is the time in which the handset synchronizes the data with the cloud service (Google Calendar) though SyncML protocol. While T_{mcs} may seem a bit higher, the phone is rather free to continue with its tasks, so not much load on it. This is possible only due to the support for push notifications at the MCM. The mobile phone just sends the request and gets the acknowledgement back. Actual

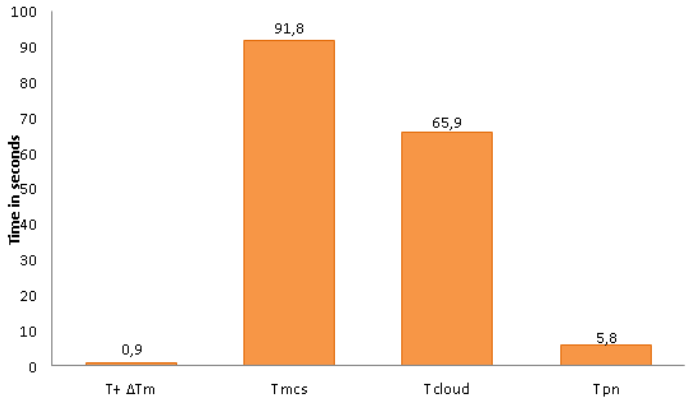

Figure 5.8: Timestamps of the application scenario

response from the cloud is sent to the mobile asynchronously. Thus the delay perceived at the mobile rather stays constant however big the T_{mcs} may be. $\cong$ is considered in the equation as there are also other timestamps involved, like the client processing at the mobile phone. However, these values will be quite small and cannot be calculated exactly.

To analyze the performance of the Zompopo application, Eucalyptus Walrus storage services are used for saving the information collected by the accelerometer. A historical set consisting in one week of accelerometer data (one file per day) was stored in a Walrus bucket (objects are stored in buckets). HTC desire phone, with a 5 megapixel color camera with auto focus and flash was considered for the analysis. It has a CPU speed of 1GHz, 576 MB of RAM and storage that can be extended up to 32GB. The application is developed based on the Android platform, compatible with Android 2.2 API or higher. Wifi connection was used to connect the mobile to the middleware. So, test cases were taken in a network with an upload rate of $\approx$ 1409 kbps and download rate of $\approx$ 3692 kbps, respectively. However, as mentioned already, estimating the true values of transmission capabilities achieved at a particular instance of time is not trivial. To counter the problem, we have taken the time stamps several times (5 times), across different parts of the day and the mean values are considered for the analysis.

The timestamps are shown in figure 5.8. The value of $T_{tr} + \Delta T_m$ is quite short ($<$ 870 msec), which is acceptable from the user perspective. So, the user has the capability to start more data intensive tasks right after the last one or go with other

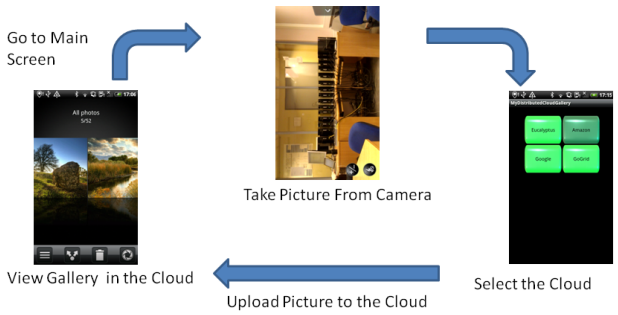

Figure 5.9: Mobile Cloud Gallery application scenario

general tasks, while the cloud services are being processed by the MCM. The total time taken for handling the cloud services at MCM, T_{Cloud} ($\sum_{i=1}^{n}(T_{te_i}+T_{c_i})$), is also logical and higher as expected (≈100 sec). The T_{pn} varies depending on current traffic of the C2DM service and has an average of ≈6 seconds.

5.2 CroudSTag: Social Group Formation with Facial Recognition and Mobile Cloud Services

The goal of the CroudSTag application is to let a person to keep in touch with people, who appear in a set of pictures, from his mobile. For example, consider a researcher who attends conferences around the world and has a set of pictures of the people with whom he had interacted at the event. The pictures are probably taken from his mobile and are stored on the cloud. The researcher later wants to create and keep connections with his acquaintances on the social network, to group them according to specific interests and to follow the groups directly from his mobile phone. The scenario can also be envisioned with any other type of the event or community that wants to keep its members in contact, something like alumni. However, for being a bit clear with the explanation, the rest of the paper is written as though it is for the researcher's case.

The CroudSTag application is realized as follows. The researcher uses the Mobile Cloud Gallery application, also developed by us, to take pictures and store them in multiple clouds. The researcher takes the picture using the mobile camera and chooses the cloud where he wants to upload the picture. Multiple clouds are considered, as the

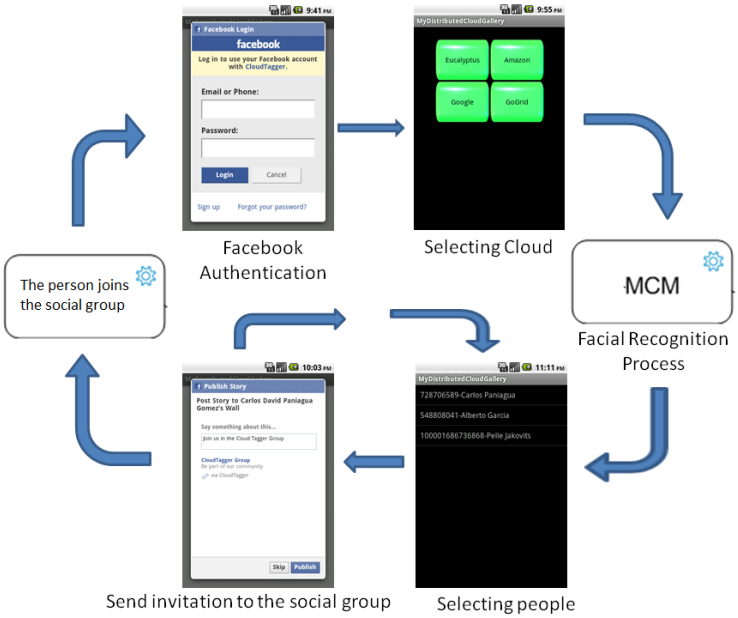

Figure 5.10: Screenshots of the CroudSTag usage scenario

researcher may store the pictures in his private cloud (something like our SciCloud, a cloud based on Eucalyptus technology with Walrus storage) when in Europe and while traveling through US or Japan he may upload them to public clouds like Amazon Simple Storage Services (S3) or GoGrid. The scenario is shown in figure. The researcher can review his pictures, stored in any cloud, at any time, in any place.

Using CroudSTag application, the researcher can later select the pictures stored in the cloud and use them to detect and to recognize his acquaintances and contact them in Facebook.com. Before starting the facial recognition process the researcher logs into Facebook.com in the application to grant access to his content (pictures, friends and tags). After the researcher is logged in, he selects the pictures he wants to use for the process, by selecting the particular cloud. Once the cloud has been selected, the mobile sends the request and the face recognition process is started on the MCM (Mobile Cloud Middleware). Face recognition is performed based on SaaS like face.com. The service is explained in detail in the next section and MCM is discussed in detail in section 4. The request from the mobile is immediately followed by an acknowledgement from the MCM and the phone is free to be used in normal way. Meanwhile, the face recognition process is performed on the cloud and the mobile is notified about the results by displaying

the recognized people in a list. The researcher then selects the persons one by one and posts a message with the invitation to join a social group in Facebook. The scenario is shown in detail in figure 5.10.

5.2.1 Cloud Services Employed in the Application

To create the CroudSTag application, several cloud services are considered, such as facial recognition services from Face.com, Facebook applications and other cloud storage and processing services. Face.com is a technology company operating in the face recognition domain, and provides several SaaS from the cloud. It offers three basic solutions for facial detection: 1) *PhotoFinder* which scans public photos in the network and suggests tags for the photos which are currently un-tagged. The aim of the application is to find pictures of a person in the network. 2) *PhotoTagger* which lets people choose albums to scan for faces grouping the people recognized in batches called screenshots. Later, PhotoTagger suggests tags for the people present in the screenshots. This batch processing is a more efficient way to tag pictures from existing albums. 3) *CelebrityFinder*, which scans pictures in Twitter looking for celebrities that have posted publicly.

In May, 2010, Face.com released its API and officially supports languages such as PHP and JavaScript. The community also has developed client libraries for Python, C#, Flash Action Script, Java and Ruby on Rails. The services for detection, recognition, and tagging are provided through its REST API; Server - http.api.face.com. The services are provided for free but with quota limitations. Face.com can also tag and recognize users from Facebook and Twitter, the two most popular social networks with billions of photos in their repositories. Face.com complies with Facebook and Twitter security policies and thus can use the credentials of the social networks. The recognition is performed on the public content in the network and the content owned by the user. It is also possible to have private sets of photos and tags, called name-spaces. However these private name-spaces need to be tagged and trained and thus reduce the number of people who can be recognized and also decreases the accuracy of the results.

The CroudSTag application also uses Facebook services to create the groups. Facebook Applications are the channel provided by Facebook, to integrate the core Facebook platform technologies such as social plug-ins, the Graph API and Dialogs. Facebook also provides several SDKs to develop applications. The JavaScript and PHP SDKs

are available for web content and the iOS and Android SDKs are available for mobile applications. The CroudSTag application uses Facebook Dialogs, which provide a consistant interface to display dialogs to users. These dialogs provide the interface to post messages in user's walls, send friend requests and applications requests.

The Graph API is the simple view to Facebook objects (people, photos, events and pages) and connections between each other. Every object inside Facebook has a unique ID and its information can be accessed through the Facebook public URL. The Graph is used to discover the relationships of one object with other objects. Relationships such as friends, events, groups, notes, albums, photos, tags, likes, etc. can be retrieved through the Graph API.

The CroudSTag application also uses the storage services from Amazon S3 and Walrus from SciCloud to store the pictures taken by the mobile phone, using native Amazon API and jets3t API, respectively. jets3t needs slight modification for handling the storage service of Walrus as it was originally meant from the community for accessing Amazon S3 services.

5.2.2 CroudSTag Performance Analysis of the Application

Once the CroudSTag application is developed, it was tested extensively for its performance. The performance model and the analysis are addressed here. figure 5.11 shows the sequence of activities that are performed during the execution of the application. Here, the total application duration i.e. the total mobile cloud service invocation time, T_{mcs} is:

$$T_{mcs} \cong T_{tr} + T_m + \Delta T_m + \sum_{i=1}^{n} (T_{te_i} + T_{c_i}) + T_{pn} \tag{5.2}$$

Where, T_{tr} is the transmission time taken across the radio link for the invocation between the mobile phone and the MCM. The value includes the time taken to transmit the request to the cloud and the time taken to send the response back to the mobile. Apart from these values, several parameters also affect the transmission delays like the TCP packet loss, TCP acknowledgements, TCP congestion control etc. So a true estimate of the transmission delays is not always possible. Alternatively, one can take the values several times and can consider the mean values for the analysis. T_m is the time taken to process the request at the middleware. ΔT_m is the minute extra latency

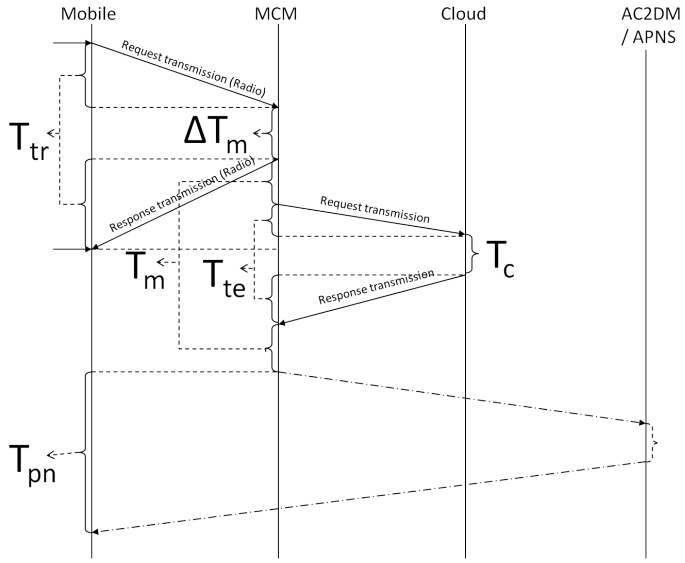

Figure 5.11: Mobile cloud service invocation cycle: Activities and timestamps

added to the performance of the MCM, as the mobile is immediately notified with the acknowledgement. T_{te} is the transmission time across the Internet/Ethernet for the invocation between the middleware and the cloud. T_c is the time taken to process the actual service at the cloud. This process is repeated several times in the CroudSTag application, as it is contacting different clouds like face.com, facebook.com, Amazon S3. Hence the sigma is considered in the equation.

Similarly, T_{pn} represents the push notification time, which is the time taken to send the response of the mobile cloud service to the device via the AC2DM. While T_{mcs} may seem a bit higher, the phone is rather free to continue with its tasks, so not much load on it. This is possible only due to the support for push notifications at the MCM. The mobile phone just sends the request and gets the acknowledgement back. Actual response from the cloud is sent to the mobile asynchronously. Thus the delay perceived at the mobile rather stays constant however big the T_{mcs} may be. $\cong$ is considered in the equation as there are also other timestamps involved, like the client processing at the mobile phone. However, these values will be quite small and cannot be calculated exactly.

To analyze the performance of the CroudSTag application, Amazon S3 storage services are used as the picture repository. 10 photos with an average of 5 faces per

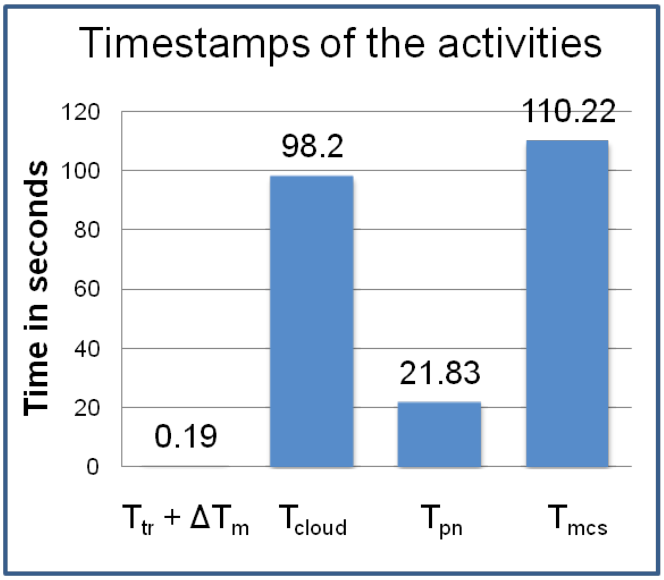

Figure 5.12: Timestamps of the application scenario

picture are stored in an S3 bucket. In S3 objects are stored in buckets. HTC desire phone, with a 5 megapixel color camera with auto focus and flash was considered for the analysis. It has a CPU speed of 1GHz, 576 MB of RAM and storage that can be extended up to 32GB. The application is developed based on the Android platform, compatible with Android 2.2 API or higher. Wifi connection was used to connect the mobile to the middleware. So, test cases were taken in a network with an upload rate of $\approx$ 1409 kbps and download rate of $\approx$ 3692 kbps, respectively. However, as mentioned already, estimating the true values of transmission capabilities achieved at a particular instance of time is not trivial. To counter the problem, we have taken the time stamps several times (5 times), across different parts of the day and the mean values are considered for the analysis.

The timestamps are shown in figure 5.12. The value of $T_{tr} + \Delta T_m$ is quite short ($<$ 200 msec), which is acceptable from the user perspective. So, the user has the capability to start more data intensive tasks right after the last one or go with other general tasks, while the cloud services are being processed by the MCM. The total time taken for handling the cloud services at MCM, T_{Cloud} ($\sum_{i=1}^{n}(T_{te_i} + T_{c_i})$), is also logical and higher as expected ($\approx$100 sec). The T_{pn} varies depending on current traffic of the C2DM service and has an average of $\approx$22 seconds.

5.3 Related work

Lot of literature exists about the accelerometer and how to use it for the recognition of human activities, the most extensive work is the one performed by Bao & Intille (2004) (37). In their experiments they used 5 biaxial accelerometers on multiple parts of the body for the recognition of activities such as eating, sitting, reading etc. Several mobile applications that enrich their functionality with the accelerometer are developed, most of them context-aware applications. However, these application do not make use of cloud services since they use accelerometer for a proactive response in real-time. For example, AAMPL (Accelerometer Augmented Mobile Phone Localization) is a mobile application that claims that localization of the mobiles can get affected slightly as GPS information is not sufficient; therefore the application enhances the localization of the user's context using the accelerometer.

Similarly, SAPM (sleep activity pattern monitoring) (38) is a health care proactive application that makes use of the accelerometer altogether with cloud services to monitor and to study in a remote way the sleep-wake cycle of elderly people staying at nursing homes. SAPM collects and stores the sensors information in a remote server (middleware analogy), later after updating the system with sleep diaries, it sends the information to the cloud for being processed by an algorithm that matches and annotates the sensor data with manual sleep diary information.

Facial recognition technologies (39) have been around for a couple of years and are one of the most promising features for the mobile users. There have been several attempts to integrate the facial recognition capabilities to mobile devices. Some manufactures such as Sony Ericsson incorporated the feature in its model X10 (40) the "Recognizer" application. This application enables the user to take a picture using the camera and recognize people present in the picture. It stores the pictures locally in the device and only recognizes people who are present in the contacts list and have a picture associated. However, it lacked the opportunity to search people in the social networks or take advantage of cloud technologies, and eventually overloads the storage resources of the phone with the pictures taken.

Viewdle (41) is another promising technology to include cloud technologies and facial recognition in mobile devices. It offers a set of solutions to provide facial recognition capabilities and intends to provide the solutions for both standalone and mobile

platforms. Currently only the desktop application in beta version has been released which enables to detect faces in photos and tries to match them with tags in the user's Facebook profile. The pictures have to be stored in the computer and during the recognition process it is possible to create a Facebook album and upload the tagged pictures. Viewdle published in its website a demo of the mobile application which allows detecting people from one picture taken with the camera and having access to his Facebook recent activity and information. Nevertheless, this application has not been released and has some problems. For example, the recognition process runs entirely in the device which might negatively impact the performance of the device in terms of battery usage, processor and time response to the user. In the context of CroudSTag application, the facial recognition service is supported by face.com. face.com only provides web or desktop solutions. Since most of the recognition process is performed on cloud, there is no dearth for resources. Face.com is currently looking for commercial applications which use its face recognition API in order to make incursion in the mobile market. We think CroudSTag may be a possible candidate.

5.4 Summary

Zompopo is an Android application that makes use of cloud services for extending the capabilities of a generic calendar within the mobile phone. Zompopo fosters the provisioning of context-aware services from the cloud, since it tries utilizing a historical set of data collected by the accelerometer (stored and processed in the cloud) for predicting activities that may help the user for scheduling his or her calendar with the activities that will be performing during the day.

CroudSTag is one such application built for the Android devices. The application takes a set of pictures from the cloud and uses the SaaS of Face.com to recognize faces and people from the pictures. The application then enables the user to send, to each recognized person, an invitation to be part of a social group and it uses Facebook for the social group creation. The social groups thus formed with people of common interests, results in sharing and collaboration relationships among the members. The paper explained the application with detailed architectural and technological choices. The performance analysis of the application shows that the social groups can be formed with significant ease and reasonable performance latencies on the devices.

6

Conclusions

Mobile domain is looking towards cloud computing with more expectation than just data synchronization, cloud resources may be used for enriching mobile applications with data-intensive processes that are time consuming and cannot be handle locally within the handset. The development of mobile applications is tied to specific cloud architecture, and thus mobile technologies are experimenting a lock in the data and application integration with other technologies. The current thesis introduced several mobile cloud services and the inherent problems in developing applications with these services. To address these problems a generic middleware framework for providing hybrid mobile cloud services is proposed. The architecture of the MCM is explained in detail and the detailed analysis of the framework is provided with an application scenario. MCM interoperability altogether with the asynchronous notification service enables the creation of powerful mashups that can invoke cloud services in a concurrent way from the handset. From this analysis we could observe that the MCM shows reasonable performance levels, thus validating the proof of concept.

While the prototype is working fine with the traditional web technologies like the HTTP and servlets, a future research will consist in extend the architecture to better suite the cellular network, by providing the access to the MCM via the XMPP protocol. XMPP is an open technology for real-time communication, which powers a wide range of applications including instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware and generalized routing of XML data. However, this requires the protocol to be implemented for the mobile platforms we have considered in the analysis; Android and iOS. Due scalability is the main concern for

centralized middleware, and the current Java implementation cannot easily scale, due the language limitations. Another improvement involves the re-implemention of the middleware using Erlang since it is more suitable for highly concurrency programs.

Another improvement involves extending the MCM to the Mobile Enterprise. A Mobile Enterprise can be established in a cellular network by participating Mobile Hosts, which act as web service providers from smart phones, and their clients. Mobile Enterprise research built a Mobile Web Services Mediation Framework (MWSMF) based on the ESB technology that helps in handling QoS, discovery and integration issues of mobile web services. The main idea consist in including the MCM as a component to the MWSMF thus providing a single interface for accessing both the mobile web services and mobile cloud services. The details will be addressed by the development of future publications.

The cloud services invocation from the handset enables a next generation of mobile applications that are not limited by storage space and processing power. These kinds of applications access the shared pool of computing resources provided by the cloud on demand, and thus are able to handle tasks that require data-intensive processing. This thesis introduces a cases of study for MCM that involves the explanation of three applications with detailed architectural (services, hardware, etc) and performance analysis. Zompopo is an Android application that makes use of cloud services for extending the capabilities of a generic calendar within the mobile phone. Zompopo fosters the provisioning of context-aware services from the cloud, since it tries utilizing a historical set of data collected by the accelerometer (stored and processed in the cloud) for predicting activities that may help the user for scheduling his or her calendar with the activities that will be performing during the day. The processing of historical data using Hadoop (MapReduce) allows finding trends that enable to context-aware applications fitting proactive responses for better adapting in the user's context.

Since Zompopo's principal aim was the consumption of cloud services from the handset for enriching mobile applications; a basic MapReduce algorithm for matching patterns is introduced. However, such algorithm will be improved in next application release for getting more precise activities prediction. Future applications development will involve the implementation of more context-aware services from the cloud that involve the use of another kind of sensors like magnetic field, in order to find patterns such as direction, presence, rotation, angle, etc.

CroudSTag is one such application built for the Android devices. The application takes a set of pictures from the cloud and uses the SaaS of Face.com to recognize faces and people from the pictures. The application then enables the user to send, to each recognized person, an invitation to be part of a social group and it uses Facebook for the social group creation. The social groups thus formed with people of common interests, results in sharing and collaboration relationships among the members.

Each performance analysis shows that this sort of applications can utilize cloud services with significant ease and reasonable performance latencies on the devices. Moreover, such applications can be executed in a concurrent way from the handset due MCM capabilities.

This thesis is conformed of a set of articles that were submitted to the following conferences, CLOUD 2011, mobiWIS 2011 (Mobile Web Information Systems 2011) and NGMAST 2011 (Next Generation Mobile Applications, Services and Technologies 2011).

7

Sisukokkuvõte

Mõlemad, pilvearvutuse ja mobiilse arvutuse suunad, on kiiresti arenenud ja paljulubavad tehnoloogiad lähitulevikus. Pilvearvutus muutub mobiilseks kui mobiiliseade võtab ühendust pilvearvutuse poolt pakutava jagatud arvutusressursiga. Mobiilsed tehnoloogiad juhivad peamiselt oma tähelepanu pilvearvutustele, põhjusel et mobiilsete rakenduste ressursi nõudlus pidevalt kasvab, olgu selleks siis kas nõudlus rohkema arvutusressursi, salvestusruumi või energiakokkuhoiu jaoks. Mobiilsed rakendused saavad kasu pilve arvutuse paindlikkusest ülesannete lahendamisel, kuid samal ajal toimivad praktiliselt reaalajas käiva teenusena.

Mobiilsete rakenduste loomine, mis kasutavad pilvearvutus teenuseid, on keeruline ja sisaldab töötamist teenuste ja API-dega erinevatelt pilvearvutuse pakkujatelt. Enamus neist API-dest ei ole omavahel ühendatud ja informatsioon, mis on arvutatud ja salvestatud ühe pilve poolt, ei ole ülekantav teistesse pilvedesse. Et ära hoida antud probleemi, vaatleb käesolev väitekiri vahevara lahendust, Mobiilse pilve vahevara (Mobile Cloud Middleware - MCM), mis haldab koostöötamise küsimusi ja kergendab keerukate teenuste koostöötamist mobiilsetel seadmetel. Töötati välja MCMi prototüüp ja üksikasjalik raamistiku jõudluse analüüsi näitab, et vahetarkvara raamistik, parandab teenuse kvaliteeti mobiilides ja aitab säilitada pehme reaalaja vastuseid.

Käesolev lõputöö on samuti realiseerinud mitu juhtumi uuringut, kasutades MCMi. Üks neist on Zompopo, Androidi rakendus, mis pakub intelligentset kalendrit, ühendades Google Calendari ja kiirendusanduri mobiilis, mis võimaldab kasutajal sättida oma ajakava päeva algusest peale, tuginedes vastavalt eelmise nätegevustele. Rakendus on detailselt lahti seletatud arhitektuuri ja tehnoloogiliste valikute kohapealt.

Rakendus kasutab MapReducei, et analüüsida kiirendusandurite andmeid, tuletades igat kõrvalekallet regulaarsest kalendri ajakavast, kasutades efektiivselt ära pilvearvutuse ressursse. Samuti on esitatud üksikasjalik rakenduse jõudluse analüüs, näidates kuidas mobiilne rakendus saab kasu pilvearvutuse kasutamisest.

Peale Zompopo prooviti lputöös veel realiseerida mõningaid teisi sotsiaalse võrgustiku rakendusi, kasutades MCMi. Sotsiaalsed võrgustikud on muutunud tänapäeval üsna populaarseteks ja inimeste sotsiaalsete gruppide kujunemise tulemusel toimub liikmete vaheline koostöö ja info jagamine. CroudSTag, mis on arendatud Androidi rakendusest, on abivahendiks mobiilsetes seadmetes ühiste huvidega sotsiaalsete gruppide kujundamisel. Antud rakendus kasutab pildiarhiivi pilve andmelaost, kasutab näotuvastuspilve teenuseid inimeste identifitseerimiseks ja moodustab tuvastatud inimestest sotsiaalsed grupid Facebookis - enamikule tuntud sotsiaalses võrgustikus. Rakendus on lahti seletatud detailselt arhitektuuri ja tehnoloogiliste valikute kohapealt. Rakenduse jõudluse analüüs näitab, et mobiilsetest seadmetest võib luua sotsiaalseid gruppe võrdlemise lihtsalt ja aktsepteeritava jõudluse näitajatega, kasutades MCM-i.

8

Future Research Directions

While the thesis findings showed that it is feasible to combine cloud services from multiple clouds and to invoke those from a mobile phone without losing tolerable user interaction with the mobile device. Several intrigues arose which needs to be answered by future research in the mobile cloud domain. In the context of synchronization, a similar approach may be use for monitoring from the handset the progress of each task meanwhile is executing in the cloud. Currently, MCM manages synchronization using the same interface that Funambol uses for synchronizing event tasks. Although, such strategy needs to provide highly scalability that cannot be achieved with the current Java implementation due language limitations. Hence the re-implementation of the solution to more powerful concurrent languages like Erlang, it needs to be study.

In terms of the mobile platform, the future research will be focus on extending the architecture to better suite the cellular network, by providing the access to the MCM via the XMPP protocol. XMPP is an open technology for real-time communication, which powers a wide range of applications including instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware and generalized routing of XML data.However, this requires the protocol to be implemented for the mobile platforms we have considered in the analysis; Android and iOS.

Similarly, from the standpoint of integration with other technologies, the interested research focus on extending the MCM to the Mobile Enterprise. A Mobile Enterprise can be established in a cellular network by participating Mobile Hosts, which act as web service providers from smart phones, and their clients. Mobile Enterprise research built a Mobile Web Services Mediation Framework (MWSMF) based on the ESB technology

that helps in handling QoS, discovery and integration issues of mobile web services (42). We are planning to include the MCM as a component to the MWSMF thus providing a single interface for accessing both the mobile web services and mobile cloud services.

Bibliography

[1] M. Armbrust et al., Above the clouds, a berkeley view of cloud computing, Tech. rep., University of California (Feb 2009).

[2] Apple, Inc, iPhone, http://www.apple.com/iphone/.

[3] Google Inc, Android, http://www.android.com/.

[4] A. Ofstad, E. Nicholas, R. Szcodronski, R. Choudhury, Aampl: Accelerometer augmented mobile phone localization, in: Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments, ACM, 2008, pp. 13–18.

[5] S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, G. Ahn, A. Campbell, Metrosense project: People-centric sensing at scale, in: First Workshop on World-Sensor-Web (WSW2006), Citeseer, 2006.

[6] T. Sohn, K. Li, G. Lee, I. Smith, J. Scott, W. Griswold, Place-its: A study of location-based reminders on mobile phones, UbiComp 2005: Ubiquitous Computing (2005) 232–250.

[7] A. Onetti, F. Capobianco, Open source and business model innovation. the funambol case, in: International Conference on OS Systems Genova, 11th-15th July, 2005, pp. 224–227.

[8] S. N. Srirama, M. Jarke, W. Prinz, Mobile web services mediation framework, in: Middleware for Service Oriented Computing (MW4SOC) Workshop @ 8th Int. Middleware Conf. 2007, ACM Press, 2007.

[9] Ericsson, Inc , Erlang programming language, http://www.erlang.org/.

[10] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in: Proceedings of the nineteenth ACM symposium on Operating systems principles, ACM, 2003, pp. 164–177.

[11] Amazon, Inc, Amazon - Amazon Web Services, http://aws.amazon.com/.

[12] Google Inc., Google Code - google application engine, http://code.google.com/appengine/ (2011).

[13] D. Nurmi, R. Wolski, C. G. G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, The Eucalyptus Open-source Cloud-computing System, 2011.

[14] S. N. Srirama, O. Batrashev, E. Vainikko, SciCloud: Scientific Computing on the Cloud, in: The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing(CCGrid 2010), 2010, p. 579.

[15] E. Cerami, S. Laurent, Web services essentials, O'Reilly & Associates, Inc. Sebastopol, CA, USA, 2002.

[16] R. Harrison, M. Shackman, Symbian OS C++ for mobile phones, Vol. 3, John Wiley & Sons Inc, 2007.

[17] I. Microsoft, Windows mobile phone, http://www.microsoft.com/windowsphone.

[18] Sony Ericsson, Inc., Blackberry phone., http://us.blackberry.com/.

[19] typica, typica - A Java client library for a variety of Amazon Web Services, http://code.google.com/p/typica/.

[20] jets3t, jetS3t - An open source Java toolkit for Amazon S3 and CloudFront, http://jets3t.s3.amazonaws.com/toolkit/guide.html.

[21] jclouds, jclouds - multi cloud library , http://code.google.com/p/jclouds/.

[22] U. Hansmann, R. Mettala, A. Purakayastha, P. Thompson, SyncML: Synchronizing and managing your mobile data, Prentice Hall, 2003.

[23] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext transfer protocol–http/1.1, Tech. rep., RFC 2616, June (1999).

[24] J. Postel, Rfc 821: Simple mail transfer protocol, Inter-Net Network Working Group, Agustus.

[25] Apache Hadoop, Apache Hadoop, http://hadoop.apache.org/.

[26] P. Saint-Andr, K. Smith, R. Tron?on, XMPP: the definitive guide : building real-time applications with Jabber, O'Reilly Media, 2009.

[27] JSON, JavaScript Object Notation, http://www.json.org/.

[28] AC2DM, Android Cloud to Device Messaging Framework, http://code.google.com/android/c2dm/index.html.

[29] iOS Reference Library, iOS Reference Library - Local and Push Notification Programming Guide, http://developer.apple.com/library/ios/.

[30] Gsm Arena, Inc, Gsm Arena - HTC Desire, http://www.gsmarena.com/htc_desire-3077.php (2011).

[31] Delta Cloud, Delta Cloud - Many Clouds. One API. No problem, http://incubator.apache.org/deltacloud/.

[32] dasein.org, dasein.org - The Dasein Cloud API, http://dasein-cloud.sourceforge.net/.

[33] zeus-mobile-cloud, zeus-mobile-cloud - A Mobile cloud computing venture that offers a new concept known as VMaaS (Virtual Machine as a Service), http://code.google.com/p/zeus-mobile-cloud/.

BIBLIOGRAPHY

[34] Q. Wang, R. Deters, Soas last mile connecting smartphones to the service cloud, in: 2009 IEEE International Conference on Cloud Computing, 2009, pp. 80–87.

[35] R. Aversa, B. Di Martino, M. Rak, S. Venticinque, Cloud agency: A mobile agent based cloud system, in: 2010 International Conference on Complex, Intelligent and Software Intensive Systems, Ieee, 2010, pp. 132–137.

[36] P. Narasimhan, Agora: mobile cloud-computing middleware, http://www.cylab.cmu.edu/research/.

[37] L. Bao, S. Intille, Activity recognition from user-annotated acceleration data, Pervasive Computing (2004) 1–17.

[38] J. Biswas, J. Maniyeri, K. Gopalakrishnan, L. Shue, P. Eugene, H. Palit, F. Siang, L. Seng, L. Xiaorong, Processing of wearable sensor data on the cloud-a step towards scaling of continuous monitoring of health and well-being, in: Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, IEEE, 2010, pp. 3860–3863.

[39] E. Hall, Computer image processing and recognition, NASA STI/Recon Technical Report A 81 (1979) 13069.

[40] Sony Ericsson Mobile Communications AB, Inc , Xperia X10 , http://www.sonyericsson.com/cws/products/mobilephones/ (2010).

[41] Viewdle, Inc, Viewdle, http://www.viewdle.com/.

[42] S. N. Srirama, M. Jarke, Mobile hosts in enterprise service integration, International Journal of Web Engineering and Technology (IJWET) 5 (2) (2009) 187–213.