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

Satish Narayana Srirama, Huber Flores, Carlos Paniagua
Institute of Computer Science, University of Tartu
J. Liivi 2, Tartu, Estonia
{*srirama, huber, paniagua*}*@ut.ee*

***Abstract*—Both cloud computing and mobile computing domains have advanced rapidly and are the promising technologies for the near future. Furthermore, the proliferation of mobile devices is fostering the emergence of ubiquitous environments, and thus the development of pervasive and context-aware applications is increasing. 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. This paper introduces 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 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.**

***Keywords*-Mobile computing; cloud computing; context-aware; accelerometer; MapReduce;**

I. 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 the users who need not have knowledge of, expertise in, or control over the cloud infrastructure that supports them. The provisioning of cloud services occurs at the Infrastructural level (IaaS) or Platform level (PaaS) or at the Software level (SaaS). 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. Some of the most popular vendors offering synchronization services are Funambol [2], Mobical.net, rseven.com and Memotoo.com.

Similarly, improvements in mobile devices, on hardware (embedded sensors, memory, power consumption, touch-screen, better ergonomic design, etc.), in software (more numerous and more sophisticated applications due to the release of iPhone [3] and Android [4] 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 in order to understand the user in multiple ways (interaction, movement, location etc.).

Furthermore, 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 [5], MetroSense [6], Place-Its [7] 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.

Mobile technologies are drawing their attention to the clouds due to the increasing demand of the applications for processing power, storage space and energy. Applications involving the use of sensors can rely on Hadoop [8] as mobile cloud service for processing (identification, recognition, classification etc.) large amounts of data. Therefore, to bring the benefits from these different domains together, we tried to build an Android application that relies on the cloud to bring its functionality on the provisioning of context-aware services. Its aim is to provide an intelligent calendar (combining Google Calendar and the accelerometer sensor) that allows the user to schedule his or her activities from the beginning of the day according to previous week's activities. The application is explained in detail with the technological and architectural choices. We have organized the rest of this paper in the following way.

Section II describes the developed application in detail, with screenshots. Section III discusses the accelerometer data used in the application and how it is processed on the cloud. Later, section IV discusses the considered architectural and technological choices. Section V provides a detailed analysis of the application. Section VI discusses the related work and Section VII concludes the paper with future research directions

II. Zompopo: Application description and screenshots

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 as mobile calendar it 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 [9] (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 III). 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

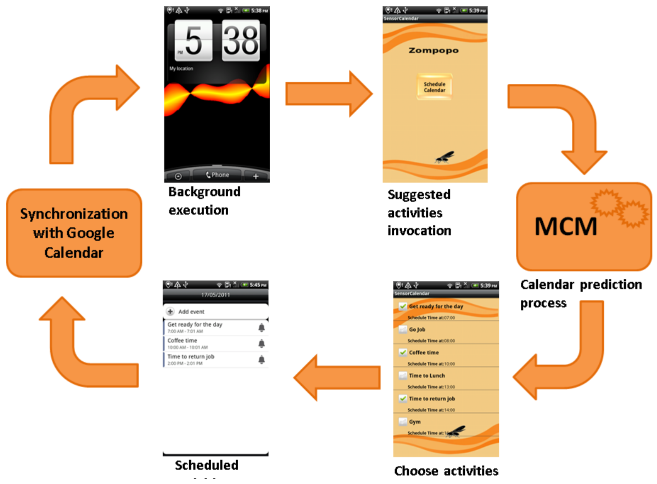


Figure 1. Zompopo application flow

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 1). MCM is discussed in detail in section IV.

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 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.

III. 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 (QoS) 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

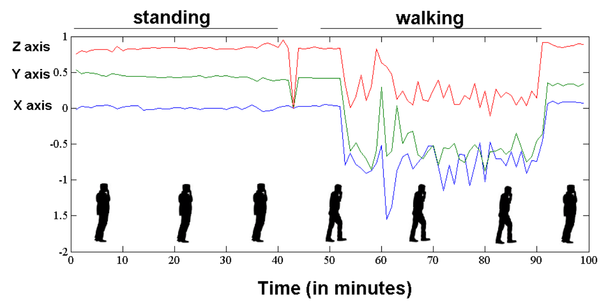
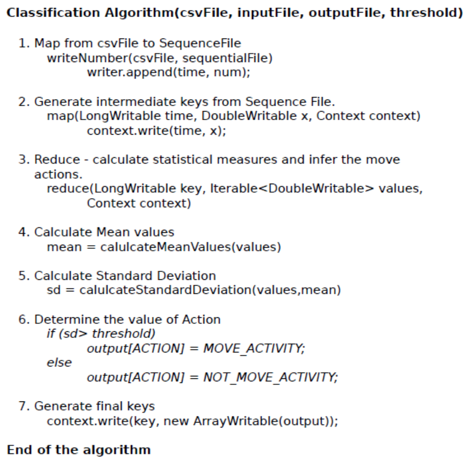


Figure 2. 3-axis readings for different activities

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 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 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 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 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 4



```
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(time, x);

  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 3. Classification algorithm using Hadoop map/reduce

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

Figure 4. Sequential file procedure

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). 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

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

Figure 5. Segregation of accelerometer data based on x axis

```
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 6. Reduce Function

the user is moving or not. The standard deviation indicates how the data is spread out over a range of values. Based on figure 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 (figure 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.

IV. Generic Middleware for Mobile Cloud Services Invocation

The direct invocation of a cloud service (such as Hadoop) is resource consuming for a mobile phone, as the operating system tends to get stuck if the computation offloading requires long waiting time for getting a response back. Moreover, meanwhile the handset is waiting for an answer, it cannot make another application call from the user, hence the mobile is not able to execute concurrent tasks. Also such waiting time is not tolerable for the user and mobile application usability perspective. Consequently, its necessary to rely on a middleware solution for getting the results asynchronously.

Several cloud services are considered in the Zompopo application, most of them are bounded by numerous constraints like cloud provider's technology choices, platform restrictions etc. Cloud providers offer proprietary APIs and routines to consume the services, e.g. Google (Gdata API suite) and JetS3t API [10] (Amazon S3 [11] and Eucalyptus [12]). Therefore, cloud interoperability is not possible and when a mobile mashup application is to be created, it has to be developed for a specific cloud provider. Moreover, the devices need to have the APIs specific to the mobile platforms like Android or iOS, and cloud vendors are generally observed to be slow in providing APIs for multiple mobile

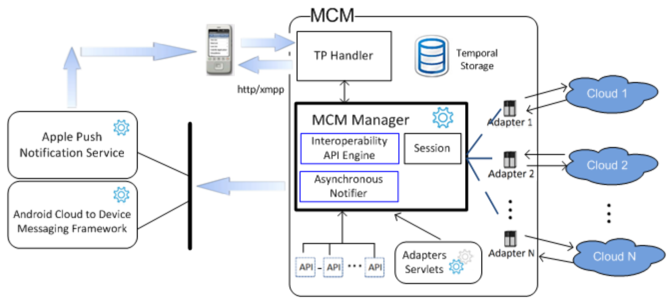


Figure 7. Architecture of the Mobile Cloud Middleware

platforms. For example, at the time of writing this paper, Amazon has just released the mobile API for Android and still in beta. To counter most of these problems in general, Mobile Cloud Middleware (MCM) has been developed.

MCM is introduced as an intermediary between the mobile phones and the clouds in the mobile cloud service invocation cycle. The architecture is shown in figure 7. 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) [13] or the Extensible Messaging and Presence Protocol (XMPP) [14] 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) [15] format, based on the application design. In case of the Zompopo application the final result is sent just once, after all the cloud services are finished. The adapter keeps the connection alive between MCM and the cloud, and monitors the status of each task running within the cloud.

When all the cloud services are completed, MCM Manager uses the asynchronous notification feature to push the response back to the handset. Since, Zompopo is an Android application, it follows the Android Cloud to Device Messaging Framework (AC2DM) [16] protocol for this task. AC2DM is a lightweight mechanism which lets to push a message into a queue of a third party notification service, which is later sent to the device. Once the message is received, the system wakes up the application via Intent Broadcast, passing the raw message data received straight to the Zompopo application.

V. 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 8 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:

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

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 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 acknowledgment. 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

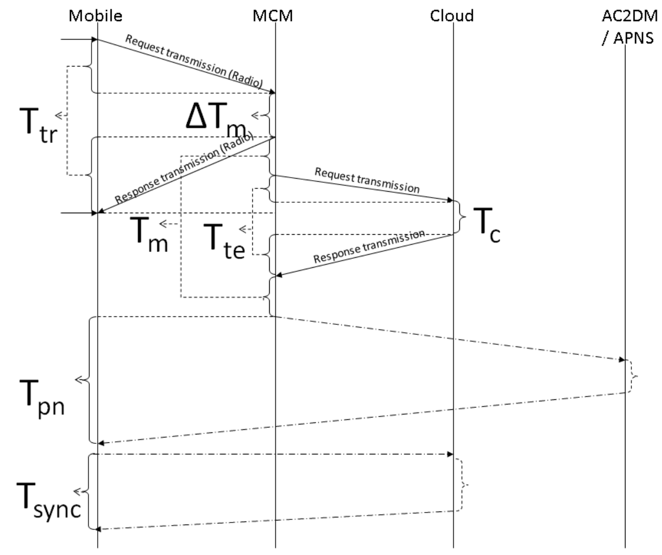


Figure 8. Mobile cloud service invocation cycle: Activities and timestamps

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 acknowledgment 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 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

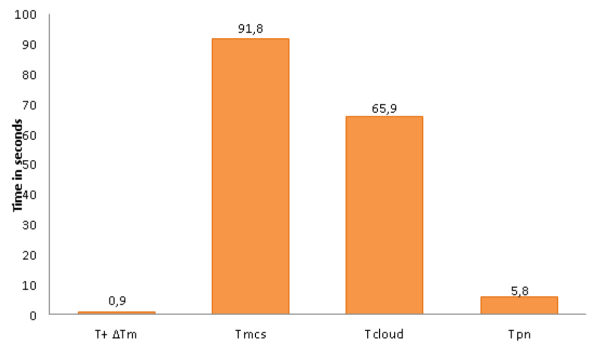


Figure 9. Timestamps of the application scenario

day and the mean values are considered for the analysis.

The timestamps are shown in figure 9. The value of T_{tr} + ΔT_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 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$6 seconds.

VI. 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) [17]. 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) [18] 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.

Middleware approaches similar to our MCM have also been addressed in the literature. MCCM (Mobile Cloud Computing Middleware) [19] is a project which involves the use of a middleware for the consumption of web services (WS) [20] in a mobile mashup WS application. However, we have observed their middleware and the API support to be tightly coupled. Cloud agency [21] is another solution that aims to integrate GRID [22], 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. Agora [23] 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. Although, we could not find any concrete implementations yet.

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) [24] that helps in offering proper QoS and discovery mechanisms for the services being provided from the smart phones [25]. MWSMF is shown to be reasonably scalable and our future research will try to add the MCM as a component to the MWSMF.

VII. Conclusions and future research directions

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. 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. The paper explained the application with detailed architectural (services, hardware, etc) and technological choices. The performance analysis of the application shows that Zompopo can utilize cloud services for processing historical data with significant ease and reasonable performance latencies on the devices. Based on the Zompopo application performance, we can think of several other application scenarios that can benefit by going cloud-aware from the mobile devices.

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. Regarding future work, we are also interested in implementing more context-aware services from the cloud that involve the data analysis from other kind of sensors like magnetic field, in order to find patterns that can be use for predicting actions, such as direction, presence, rotation, angle, etc.

Acknowledgment

The research is supported by the European Social Fund through Mobilitas program and the European Regional Development Fund through the Estonian Centre of Excellence in Computer Science.

Zompopo is a flying ant that appears only in May specifically in the Central American region. The application is called Zompopo due to its predictive behavior.

References

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

[2] A. Onetti and 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.

[3] Apple Inc, "IPhone," http://www.apple.com/iphone/.

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

[5] A. Ofstad, E. Nicholas, R. Szcodronski, and 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.

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

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

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

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

[10] jets3t, "jetS3t - An open source Java toolkit for Amazon S3 and CloudFront," http://jets3t.s3.amazonaws.com/toolkit/guide.html, 2011.

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

[12] D. Nurmi, R. Wolski, C. G. G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-source Cloud-computing System," 2011, uRL last visited on 5th Nov 2010.

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1," RFC 2616, June, Tech. Rep., 1999.

[14] P. Saint-Andr, K. Smith, and R. Troncon, *XMPP: the definitive guide : building real-time applications with Jabber*. O'Reilly Media, 2009.

[15] JSON, "JSON," http://www.json.org/.

[16] Google Inc., "Google code labs - Android Cloud to Device Messaging Framework," http://code.google.com/intl/es-ES/android/c2dm/index.html.

[17] L. Bao and S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive Computing*, pp. 1–17, 2004.

[18] J. Biswas, J. Maniyeri, K. Gopalakrishnan, L. Shue, P. Eugene, H. Palit, F. Siang, L. Seng, and 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.

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

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

[21] R. Aversa, B. Di Martino, M. Rak, and 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.

[22] F. Berman, G. Fox, and A. J. Hey, *Overview of the Book: Grid Computing–Making the Global Infrastructure a Reality*. Wiley Online Library, 2003.

[23] P. Narasimhan, "Agora: mobile cloud-computing middleware," http://www.cylab.cmu.edu/research/projects/2010/agora.html.

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

[25] S. N. Srirama and M. Jarke, "Mobile hosts in enterprise service integration," *International Journal of Web Engineering and Technology (IJWET)*, vol. 5, no. 2, pp. 187–213, 2009.