Misco: A System for Data Analysis Applications on Networks of Smartphones using MapReduce

Theofilos Kakantousis*, Ioannis Boutsis*, Vana Kalogeraki*, Dimitrios Gunopulos†, Giorgos Gasparis†, Adam Dou‡
*Athens University of Economics, Greece and Business, {t.kakantousis, mpoutsis, vana}@aueb.gr
†University of Athens, Athens, Greece, {g.gasparis, dg}@di.uoa.gr
‡Google, USA, adam.dou@gmail.com

***Abstract*—The recent years have seen a proliferation of community sensing or participatory sensing paradigms, where individuals rely on the use of smart and powerful mobile devices to collect, store and analyze data from everyday life. Due to this massive collection of the data, a key challenge to all such developments, is to provide a simple but efficient way to facilitate the programming of distributed applications on the embedded devices. We will demonstrate a novel system that provides a principled approach to developing distributed data clustering applications on networks of smartphones and other mobile devices. The system comprises three components: (a) a distributed framework, implemented on mobile phones that eases the programmability and deployment of applications on the devices using simple programming primitives, (b) a data gathering component that tracks the movement of wireless device users and collects sensor data (*i.e.,* GPS and accelerometer sensor data), and (c) a distributed data clustering algorithm that allows users to combine their individual data, that is distributed and energy efficient. Using a road traffic monitoring application we demonstrate how MISCO can efficiently identify anomalies in the road surface conditions and illustrate that our system is practical and has low energy and resource overhead.**

I. Introduction

Portable electronics such as smart-phones, PDAs and mobile entertainment units have become very popular over the past few years, with smart-phones being the fastest growing segment of the mobile phone market. In the early 2012 period, it is reported that more than 104 million people in the U.S. alone own smart-phones, 14 percent higher than 4 months prior [1]. These devices are outfitted with a wide array of sensing capabilities such as GPS, WiFi, accelerometers, microphones, cameras and accelerometers, which have introduced new and more efficient ways of communication. The processing power, networking capability and resources available on these devices are also increasingly rapidly, newer smart-phones feature 2 MIPS processors, 512MB of memory and tens of gigabytes of storage.

With smart phones becoming a platform of choice, a number of applications have emerged in a wide variety of domains ranging from personal life, to travel and work. Examples include traffic monitoring systems for real-time delay estimation and congestion detection, location-based services such as personalized weather information, location-based games or receiving spatial alarms upon the arrival to a reference time point, as well as social networking applications for sharing photos and personal data with family and friends [8] [5]. Providing a simple way to facilitate the programming and deployment of distributed data clustering applications on the mobile devices presents significant challenges to the mobile system design.

Consider the example of a road traffic monitoring application. Prior work in this area has required the deployment of dedicated equipment on the vehicles (e.g., using navigators) or the tracking of mobile phones by service providers. In these systems, people are generally aware of traffic flows and alternative routes, but it is much more difficult to communicate dynamically unexpected conditions, such as traffic congestion over short time periods, accidents, obstacles or hazards. The road traffic monitoring application introduces several challenges including gathering sensor data in real-time, efficiently identifying unexpected events such as traffic conditions or road surface anomalies and communicating these events to other users when certain conditions arise or when they are needed by the other users. To fully take advantage of this sensing infrastructure, a simple, but versatile system which eases the development and deployment of software must be created. We will show that our system can be efficiently used to address these challenges.

We have developed a system that supports the development of distributed data clustering applications on networks of smartphones. Our system depicts several features: First, it provides a distributed framework, based on the MapReduce programming model, the MISCO framework, that eases the programmability and deployment of applications on the mobile devices using simple programming primitives. A *data gathering component* runs on the mobile devices that tracks the movement of wireless device users and collects sensor data (*i.e.,* GPS and accelerometer sensor data). As users move around to perform their daily activities, sensor data paired with location information is gathered periodically and stored locally on the phones.

To complement the data gathering component we have developed a distributed data analysis component. Our goal is to develop distributed data analysis techniques: distributed techniques have important fault-tolerance characteristics, and the use of our Misco framework eases the programming of sophisticated distributed mechanisms. In this paper we demonstrate an *distributed clustering algorithm* that operates on the

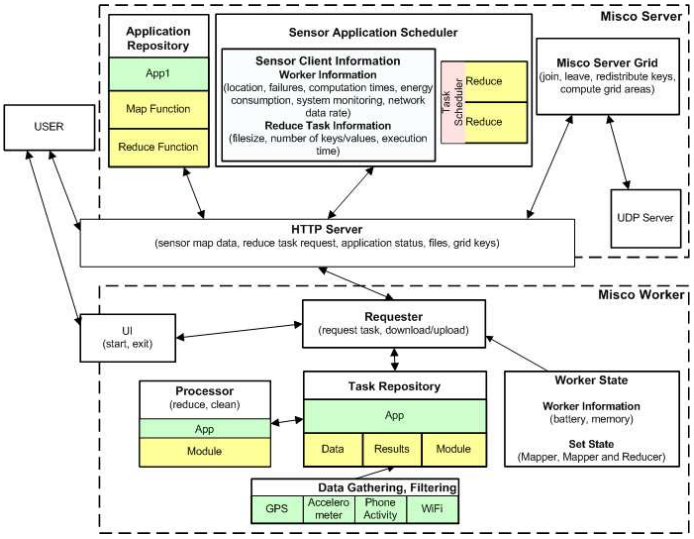

Fig. 1. The architecture of our system on the N95 smartphones.

data that are collected by the smartphones. Clustering is one of the fundamental data analysis operations, and it is useful in several domains of science, including biology, economics, and engineering. The algorithm is a MapReduce implementation of the well known K-Means clustering algorithm. Our implementation is designed to minimize the cost of data transmissions, and also protect the privacy of individual users by letting each smartphone share only aggregate information. We stress here that our goal is not so much to illustrate the advantages of clustering in general and distributed K-Means clustering in particular; our goal is to demonstrate that sophisticated and widely used data analysis techniques can be efficiently and easily implemented in the Misco framework. Finally, the *user interface* displays a map of the region and shows areas where there is good connectivity based on the resulting centroids from the distributed clustering.

Using a road traffic monitoring application we demonstrate how the system can efficiently identify anomalies in the road surface conditions. Our detailed experimental results over our testbed of Nokia N95 8GB smartphones, illustrate that our system is practical and has low energy and resource overhead.

II. System Architecture

In this section, we present the main components of the system implementation (shown in Figure 1). The system consists of a collection of software components running on Nokia N95 mobile phones and a MapReduce server which is used to coordinate the distributed processing on the phones. The software on the phones performs sensing, gathering of GPS and accelerometer data, distributed processing of the sensed data and displaying these results to the user GUI.

The MapReduce Programming Framework: The foundation for this work is a distributed programming model, based on the Misco framework. Misco [2] uses the MapReduce programming model to simplify the programmability of the applications on the phones. MapReduce is a flexible and powerful distributed processing framework that was originally designed to automatically process large datasets using a large number of worker nodes. The main insight of the MapReduce programming model is that a large computation is split into a number of smaller tasks; these tasks are independent of each other and can be assigned on different worker nodes to process different pieces of the input data in parallel. MapReduce defines two functional language primitives: *map* and *reduce*. The map function is applied, in parallel, to pieces of input data and produces intermediary $< key, value >$ pairs. The pairs are then partitioned by key (e.g. $hash(key)$ MOD R) and each partition is passed into a reduce function which produces further results. The MapReduce framework's support for the weak connectivity model of computations across open networks makes it very appropriate for a mobile network setting. We have implemented the MapReduce framework on the Nokia N95 mobile phones. Using such a framework allows us to concentrate on the algorithm without worrying about the difficult low-level issues such as code distribution, parallelization and inter-device communication.

Data Gathering Module: The *data gathering* module on the phone is responsible for collecting data from the sensors on the phones as the users move. The data is stored locally in local profiles and will be provided to the rest of the system for distributed processing as well as data analysis. The collection on the phone involves two steps: (i) GPS readings to provide geographical location of the phone, and (ii) accelerometer readings to identify road conditions.

The data gathering component on each phone performs a series of local queries to obtain a timeseries of data points. To get the current location of the phone, the GPS readings are provided through local calls to the Nokia GPS daemon. The GPS receiver are obtained via the positioning module returning data at a user defined rate which in our case was set to one second. The accelerometer sensor provides a vector of readings that can be used to sense orientation (because direction of weight changes), vibration, shock, and falling (e.g., in the case where the acceleration changes). In order to obtain data from the accelerometer the sensor module was used, particularly the AccSensor type while the Map function acts as a callback to manipulate these data.

Data Clustering: Our distributed clustering algorithm [3] is a distributed implementation of the K-Means Clustering algorithm [4] which performs data clustering on data collected at the phones to determine geographical areas of good connectivity. The distributed clustering algorithm works in the following phases: (a) collection of local data to determine areas of good connectivity within the mobile phone's proximity, and (b) exchange of data points between the phones to refine the good connectivity areas and obtain knowledge of other areas across the greater geographical region. Our MapReduce implementation of the K-means algorithm minimizes the data movement by only exchanging centroints between the phones and not the actual data, thus the energy wastage is proportional to the number of hotspots (K) and not to the number of data. This also provides a degree of user privacy as the exact

Fig. 2. The user interface displaying the anomalies detected in the real-time traffic monitoring application.

location information of the individual mobile phones is not revealed to any user.

Real-time Road Traffic Monitoring Application: We have implemented a road traffic monitoring application (Figure 2) that collects GPS and accelerometer sensor data to monitor traffic conditions such as sudden stops, breaking and potholes. The accelerometer sensor is used to record the forces applied on the phone in the 3-dimensional space while moving inside the owners vehicle and the GPS receiver is responsible for obtaining the geographical position and speed of the device. The sensor data is collected and stored locally on the phones and analyzed to identify the different types of anomalies. The anomalies extracted could be annotated on a map, thus allowing the user to search for alternative routes while driving and avoiding chaotic roads and intersections. We will show that we can create an inexpensive infastructure that runs on top of smartphones that can be efficiently used to detect and report such anomalies in the road conditions.

III. Implementation

Our MapReduce implementation consists of a *Master Server* and a number of *Worker Nodes*. The system collects and processes raw data from the data gathering component on the phones and generates results for the graphical user interface (GUI). The Master server keeps track of the geographic locations of the Worker nodes, maintains intermediary results related to the data clustering and determines how tasks should be assigned to the Workers nodes. The MapReduce system is designed to operate on the Nokia N95 8GB smart-phones [6] which support a beta implementation of Python 2.5.4 called PyS60. Although we target this specific phone, our system can run on any Python enabled system.

We chose Python for the implementation of our system modules due to its advantages of expressiveness, ease-of-development and portability across many devices. In order to obtain data from the accelerometer, the sensor module was used, and in particular the AccSensor type while the Map function acts as a callback to manipulate these data. Necessary information from the GPS receiver are obtained via the positioning module returning data at a user defined rate which in this case is one second. User interaction with the phone is detected using the e32 and telephone modules. The inactivity function from the e32 module is used to measure the time period in seconds since the last time a button on the phone was pressed. The second module uses a callback function to store the mobile phone's state and depending on its value the device filters the data to be sent to the server.

Another important aspect in the system was to enable the users take and receive phone calls. User interaction with the phone is detected using the e32 and telephone modules. The inactivity function from the e32 module is used to measure the time period in seconds since the last time a button on the phone was pressed. The second module uses a callback function to store the mobile phone's state and depending on its value the device filters the data to be sent to the server.

Worker: The worker component of the MapReduce system runs on the phones and is responsible for processing map and reduce tasks and returning the results to the server. The input data for the tasks is the data which has been collected by the data gathering component. In particular, the data gathering component collects a timeseries of tuples in the form $< latitude, longitude, X, Y, Z, speed, time >$ where $latitude$, $longitude$, $speed$ and $time$ are provided by the GPS receiver and represent the location and speed of the mobile phone when the event was recorded, while X, Y, Z refer to the phones movement axes, obtained through the accelerometer sensor. The worker consists of a *Requester* component that is used for communicating with the server to request tasks, upload and download data and to trigger the local execution of the tasks, and a *Task Repository* component that keeps track of the tasks locally exected. The communication between the server and worker is accomplished through the *HTTP server* using a special URL where the server listens for requests. Each task is characterized by the location of the module containing the Python functions for the map or reduce operations. The worker downloads and places this module locally in its *Task Repository* component. The *Processor* then uses Python's dynamic import ability to extract and run the functions in the downloaded task module. Finally, the *Worker State* component is used to maintain local statistics regarding the resource utilization on the phones.

The distributed clustering application processes the road traffic data that has been gathered locally on the phones by the workers. The data clustering algorithm is implemented through the reduce tasks that is also executed on the phones. During this phase, the input from different map tasks is used as input to the reduce task. This will produce the set of all road anomalies detected by the phones.

Server: The server component of the MapReduce system runs on a server machine with a publicly accessible IP address. It is responsible for keeping track of the distributed clustering application and assigning tasks to workers. The server is multi-threaded, spawning a new thread to handle incoming worker connections and user requests. It uses $RLocks$ for synchronizing critical sections. The user interfaces with the server through a HTML interface which allows to check the application progress. The server comprises a *Sensor Application Scheduler* component to schedule the execution of applications tasks, an *Application Repository* component that

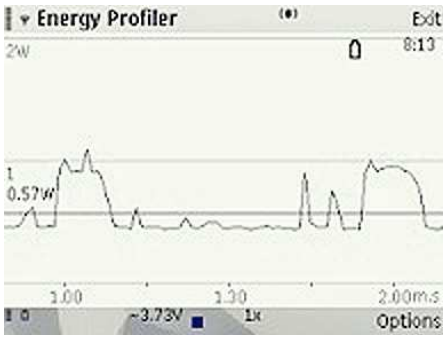

Fig. 3. Energy measurements.

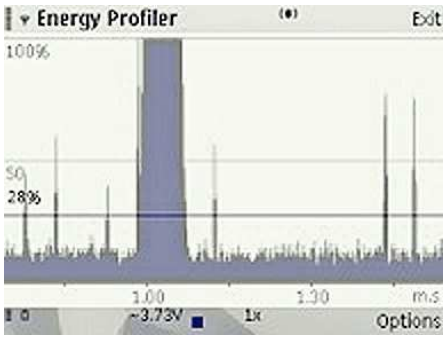

Fig. 4. CPU measurements.

keeps track of application modules and intermediary data, a Misco Server Grid that is responsible to keep track of the geographic location of the phones, and an *HTTP Server* that serves as the main communication between the workers and the server. The HTTP server is responsible for receiving requests, displaying application statuses to the user via a web UI and handling the downloading and uploading of data files.

IV. Demonstration Settings

Equipment: We will be using a testbed of Nokia N95 8GB smart-phones. These phones are equipped with ARM 11 dual CPUs at 332 Mhz, supports wireless 802.11b/g networks, bluetooth and cellular 3g networks, 90 MB of available main memory and 8 GB of local storage. The N95 runs the Symbian OS (S60 3rd Edition Feature Pack 1) and our application is implemented as a Python module which runs on PyS60 (a beta implementation of Python 2.5.4). Figure 2 shows a graphical interface of the operation of our system.

For our MapReduce server, we will use a laptop with an Intel Core 2 Duo 1.83 Ghz with 1 GB of ram. The server is implemented in Python and the server is running Python 2.5.2 on Windows XP. The laptop will have a 100MBit wired connection to a Linksys 802.11g router, which the phones will connect to wirelessly.

Demo Plan: We will demonstrate the data gathering, distributed clustering and user interface of the phones. For the data gathering, the user will be able to trigger a data gathering phase on the phone and the phone will generate GPS data and detect Bluetooth and WiFi connections in the area. For distributed clustering, the phones will be preloaded with data traces to use as input for the distributed clustering algorithm. The MapReduce server running on the laptop will display the status of the distributed clustering as it is being processed. Finally, for user interface, the user will be presented with the results of the data clustering on the trace data. Users will be able to use the phone's GUI to locate the closest regions of good connectivity and view the connectivity of different areas. The user interface provides a map based interface which displays areas of high connectivity using colored overlays. It also shows the user's current location and provides the closest locations of good connectivity.

We have evaluated the energy and resource overheads of the sensor monitoring and data clustering application on our system. We use the Nokia Energy Profiler [7] to measure the power, memory and CPU usage on the phones. The Profiler can be set to record measurements every quarter second and provides an API to start and stop measurements. Figures 3 and 4 illustrate the energy and CPU usage measurements on the phones during the operation of the real-time road traffic monitoring application. Our experiments in evaluating the energy and CPU usage on the phones shows that our system is efficiently implemented on the smartphones and depicts small overhead.

Acknowledgment

This research has been supported by the European Union through the Marie-Curie RTD (IRG-231038) project, a Hellenic Republic Ministry of Education, Lifelong Learning and Religious Affairs Thalis DISFER project, the MODAP project, and by AUEB through a PEVE project.

References

[1] "Comscore reports February 2012", http://www.comscore.com.
[2] A. Dou and V. Kalogeraki and D. Gunopulos and T. Mielikinen and V. Tuulos, "Misco: A MapReduce Framework for Mobile Systems", PETRA, Samos, Greece, June 2010.
[3] A. Dou, V. Kalogeraki, D. Gunopulos, T. Mielikinen, V. Tuulos, S. Foley, C. Yu, "Data Clustering on Networks of Mobile Smartphones", SAINT 2011, Munich, Germany, July 2011.
[4] J. B. MacQueen, "Some Methods for Classification and Analysis of MultiVariate Observations", Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability, pages 281-297, volume 1, year 1967.
[5] E. Miluzzo and C. Cornelius and A. Ramaswamy and T. Choudhury and Z. liu and A. Campbell, "Darwin Phones: the Evolution of Sensing and Inference on Mobile Phones", Mobisys 2010, June 15-18, 2010, San Fransisco, CA.
[6] Nokia, N95 8GB Device Details, http://www.forum.nokia.com/devices/N95_8GB
[7] Nokia Energy Profiler, http://www.forum.nokia.com/ main/ resources/ user experience/ powermanagement/ nokia energy profiler/
[8] A. Thiagarajan and L. Ravindranath and K. LaCurts and S. Madden and H. Balakrishnan and S. Toledo and J. Eriksson, "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones", SenSys 2009, pages 85–98, Berkeley, California.