Demo: BlueMountain: An Architecture to Customize Data Management on Mobile Systems

Sharath Chandrashekhara, Taeyeon Ki, Kyungho Jeon, Karthik Dantu, Steven Y. Ko
Department of Computer Science and Engineering
University at Buffalo, The State University of New York
{sc296, tki, kyunghoj, kdantu, stevko}@buffalo.edu

ABSTRACT

BlueMountain is a system that enables building pluggable data management solutions which can be linked with any Android app at runtime, without requiring any modifications to the Android platform. BlueMountain simplifies the app development, provides flexibility to end users, and works with existing apps.

1. MOTIVATION AND OVERVIEW

Modern apps on mobile systems employ sophisticated data management solutions, including local file and database storage as well as cloud services for user/app data. However, data management in mobile apps does not provide enough flexibility to either developers or users.

To address this problem, we have designed a pluggable data management solution for modern mobile platforms like Android. Our goal is to allow data management mechanisms and policies to be implemented independently of core app logic and provide users a higher control over their data. To achieve this, we envision two kinds of apps. The first category is regular apps that one can currently find in app stores such as Google Play. The second category is what we call *data management apps*. These apps are meant to be *pluggable* and always used in conjunction with regular apps for the purpose of managing their data. A regular app delegates all its data management decisions to a data management app. Depending on their needs, the users will choose a data management app to use for each regular app. For example, a user in this ecosystem could install a regular note-taking app with a data management app designed to extend battery life while syncing notes to cloud. At the same time, another user could install the same note-taking app, but with a data management app that syncs all notes with the user's personal devices.

To realize the aforementioned separation, we propose BlueMountain, which has three distinct features: (i) It defines a clean interface for data management apps, which mimics the interface for Android's files, database, and key-value storage. (ii) It enables dynamic binding between a data management app and a regular app, so that a user can choose which data management app to use for each regular app. (iii) It automatically transforms existing apps to leverage BlueMountain's functionality via Java bytecode instrumentation [5]. Figure 1 shows a visualization of this workflow.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

MobiSys'17 June 19-23, 2017, Niagara Falls, NY, USA
© 2017 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-4928-4/17/06.
DOI: http://dx.doi.org/10.1145/3081333.3089327

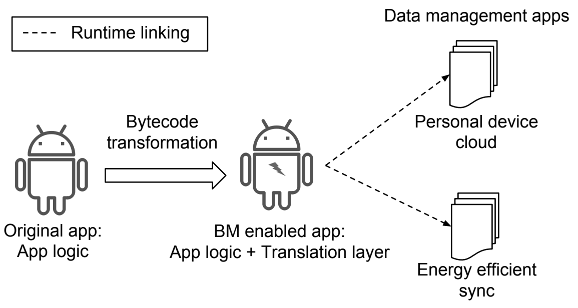

Figure 1: BlueMountain Workflow Example

We have built BlueMountain system for Android and implemented a layer that translates the Android Storage APIs—files, databases and key-value storage to our interface. We have also built five data management apps. Two of these apps (inspired by PSCloud [3] and QuickSync [4]) are designed to manage files while the other three are designed to manage databases using various cloud syncing strategies. We evaluate our system by measuring the overhead using micro-benchmarks as well as end-to-end performance with five data management apps described above. We show that our system is practical and the overhead remains less than 10% in most of the cases and goes up to 30% in the worst case. Lastly, we show that our approach *supports existing apps* by automatically transforming 400 apps downloaded from Google Play to leverage our framework and verifying the correctness of their functionality. We have measured the increase in code size, energy, and heap usage due to instrumentation and show that the overhead is minimal.

2. DEMO

Our demo [1] shows an app downloaded from Google Play, Ex., 'Photo talks' [2], instrumented with BlueMountain framework and working with one of the data management app that we have built (Ex. automatic syncing of photos generated by 'Photo talks' with all of user's devices and Dropbox).

3. REFERENCES

[1] Demo. https://goo.gl/QthkKa.
[2] Photo talks: speech bubbles. https://goo.gl/8CRheK.
[3] S. Bazarbayev et al. Pscloud: a durable context-aware personal storage cloud. In *HotDep 2013*.
[4] Y. Cui et al. Quicksync: Improving synchronization efficiency for mobile cloud storage services. In *MobiCom 2015*.
[5] T. Ki et al. Reptor: Enabling api virtualization on android for platform openness. In *MobiSys 2017*.