BlueMountain

Enabling Automated, Rich, and Versatile Data Management for Android Apps

Sharath Chandrashekhara, Kyle Marcus, Rakesh G. M. Subramanya, Hrishikesh S. Karve, Karthik Dantu and Steven Y. Ko

Reliable Mobile Systems Lab
http://www.nsr.cse.buffalo.edu

University at Buffalo The State University of New York

Mobile Apps - State of Art

- Use local and cloud storage; rich forms of interaction, backup, sharing etc.
- Large companies use their own cloud; smaller developers use public cloud
- Too many cloud providers, no standard interface

Life as a Developer

- Several design choices
- Consistency models, interface and semantics
- Tangential and repetitive
- Binds an app to a particular cloud

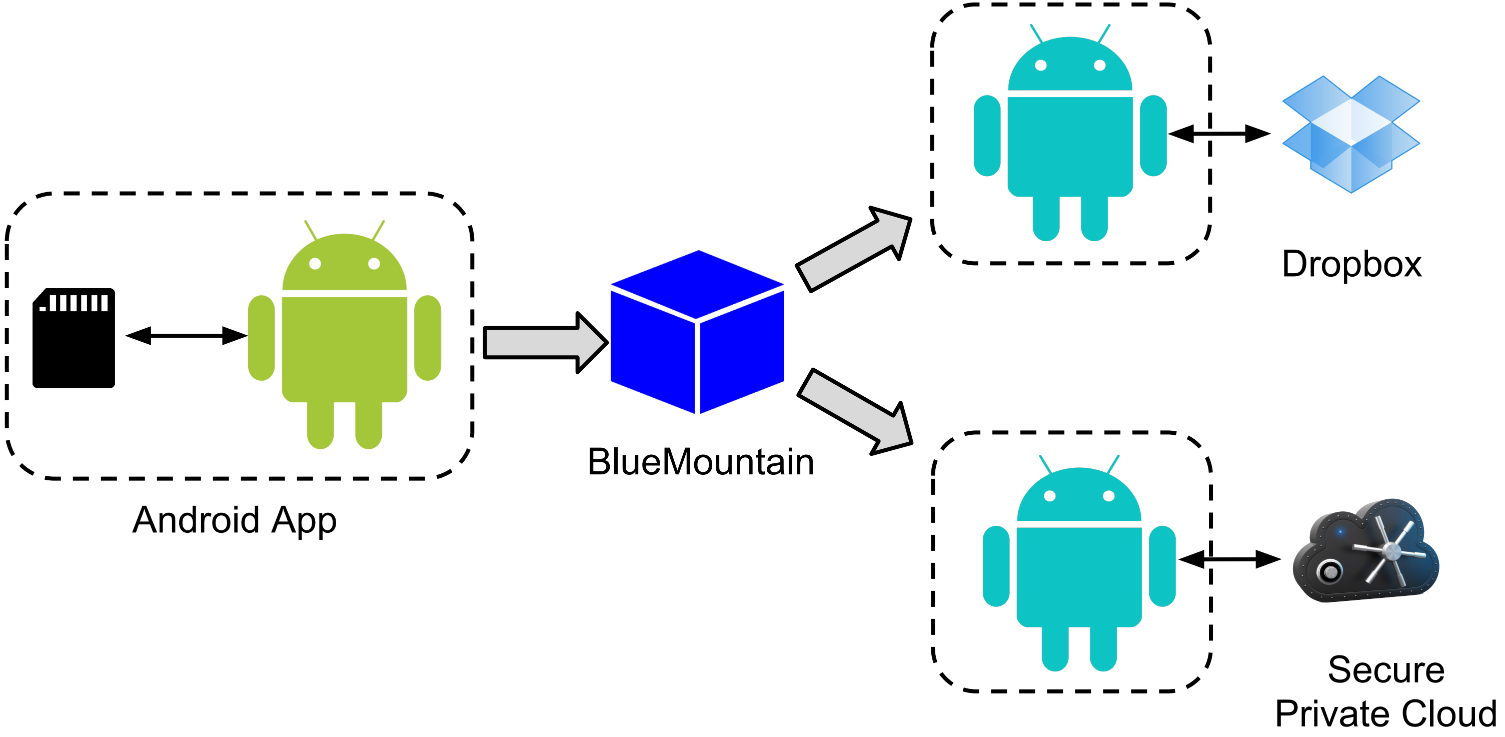
➪ *Developers want to reduce development time and provide more flexibility to users*

Life as a User

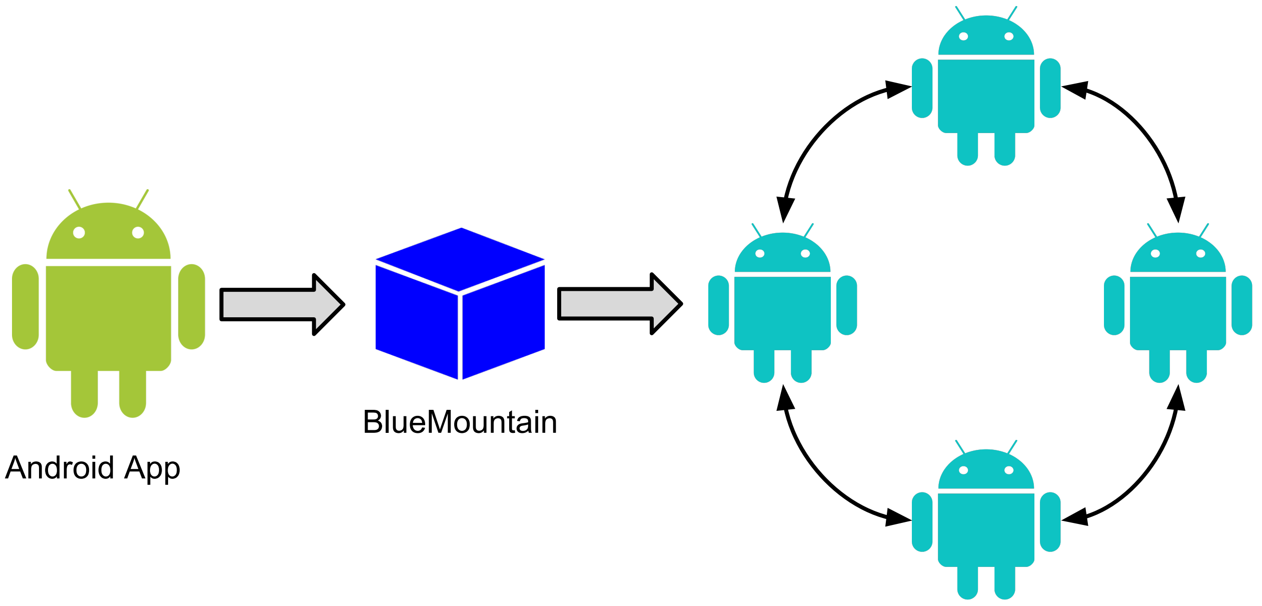
- Constrained by the developer’s decisions
- Privacy concerns when data is moved to cloud
- Has to contact the developers for any customization

⇨ Users want flexibility and control

BlueMountain: Backup Scenario



BlueMountain: P2P Sharing Scenario



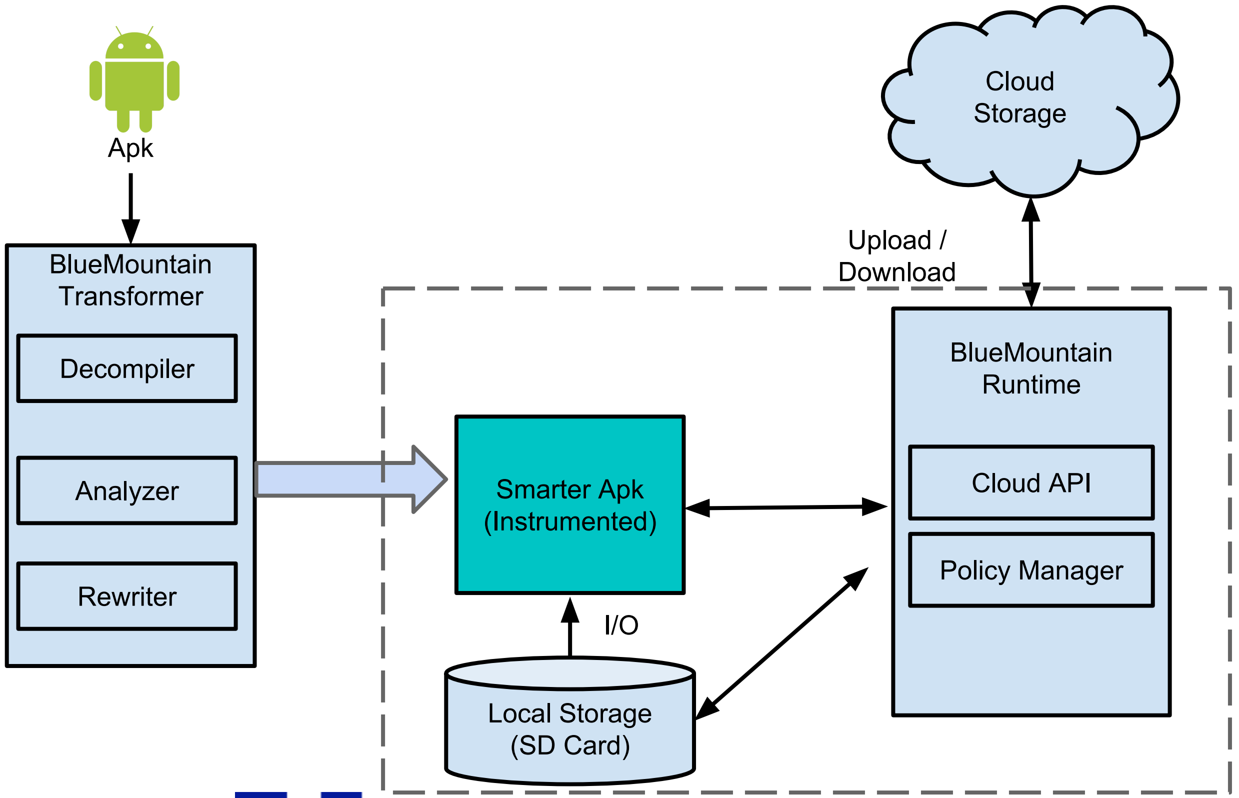
BlueMountain Goals

- **Reduce development effort:** Focus on app logic; treat all storage operations as local
- **Automatically transform apps:** Enable richer forms of data interaction
- **Flexibility:** Customize without access to source code
- **Post-development cycle:** Works with existing apps; no modifications to the Android platform for ease of deployment.

BlueMountain

- A system that automatically integrates cloud storage services with Android apps
- Main components
  - **App Transformer:** Analyses and rewrites app binaries by virtualizing the storage calls and enables richer data interactions
  - **Runtime:** Runs as a regular app; and interacts with the cloud and manages policies

BlueMountain Architecture



Challenges: Storage Virtualization

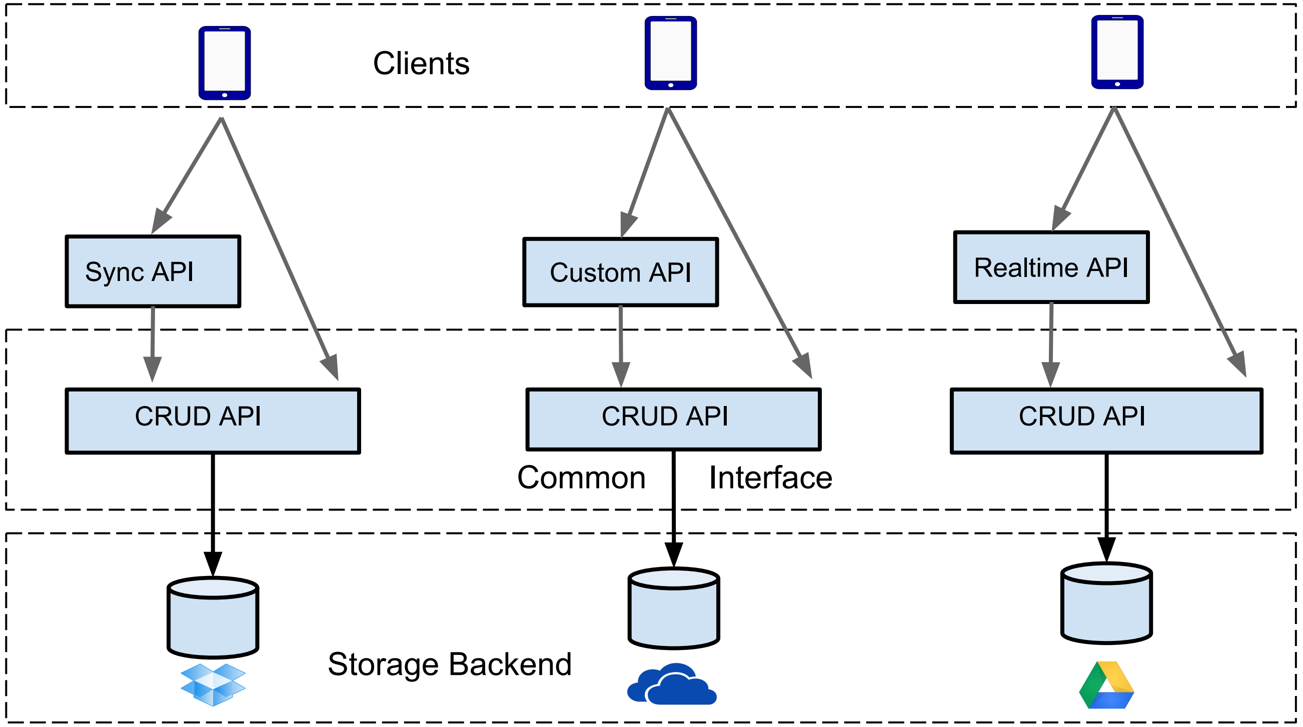
- Can we virtualize storage calls?
- Android options:
  - Files
  - Database
  - Key/Value

Storage Call Virtualization

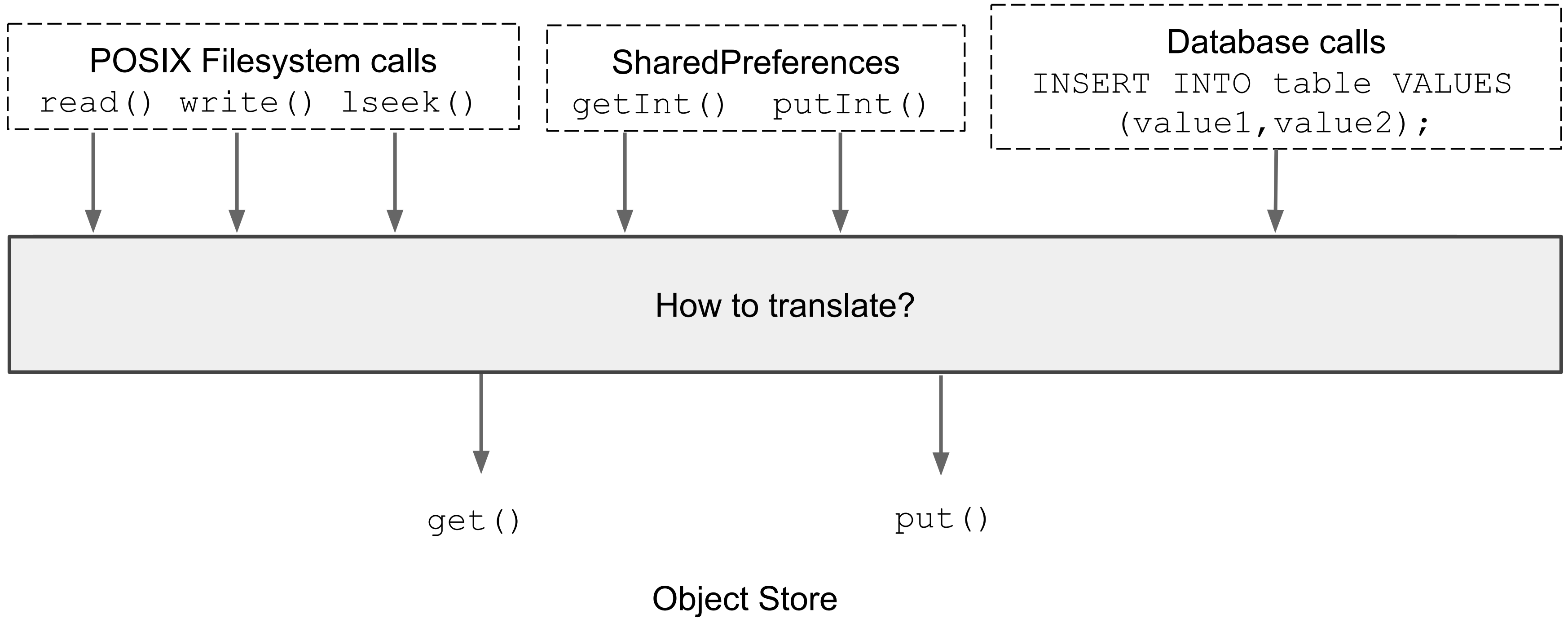
```
public class MyFileOutputStream extends
FileOutputStream {
  public int write (Bytes b) {
  //Overriding
  }
}

public class main {
  public static void main (String args[]) {
    Bytes b = 10;
    MyFileOutputStream obj = new
      MyFileOutputStream ();
      myWrite (obj, b);
  }
  public static void myWrite
    (FileOutputStream obj, Bytes b) {
    obj.write (b);
  }
}
```

- Need to statically identify all possible storage options and their APIs
- More challenging than search and replace because of polymorphism

Challenges: Cloud APIs



Challenges: Interface



Challenges: Semantics

- Handling concurrent updates
- Most clouds provide only eventual consistency
- Timing differences between local and cloud
- Time-bound eventually-consistent model?
- Getting additional inputs from the developers?

Related Work

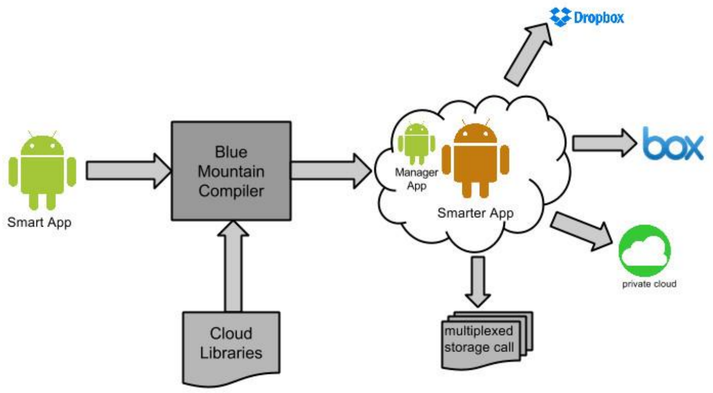
- **Viewbox, Simba** - Fault tolerance and consistency guarantees
- **Cimbiosys** - Selective sharing of files
- **Procrastinator** - Rewriting the binary
- **MetaSync** - Enhances cloud services
- **CloudRail** - Unified API

Conclusion & Future Work

- Initial vision for BlueMountain - *storage virtualization* and *cloud integration*
- Developed a prototype to demonstrate these features
- Future work
  - Full implementation
  - Resolving interface and semantic mismatches
  - Analysing and categorizing the apps of Play Store and corporate apps which store data on their private cloud

BlueMountain

Enabling Automated, Rich, and Versatile Data Management for Android Apps



Sharath Chandrashekhara, Kyle Marcus, Rakesh G. M. Subramanya,
Hrishikesh S. Karve, Karthik Dantu and Steven Y. Ko
{sc296, kmarcus2, rakeshgu, hkarve, kdantu, stevko}@buffalo.edu
Reliable Mobile Systems Lab
http://www.nsr.cse.buffalo.edu

University at Buffalo *The State University of New York*