Virtual Meow
Virtual File System on Android

Kam Cheung Ting

Tian Xia

Ruogu Zeng

Background

- Growing Demand for Phone Storage
  - Faster Mobile Network
  - More Videos, Images, Musics
  - Larger Size Application and Data Package
    - Ex. 3D Games on Mobile Phone

Problem

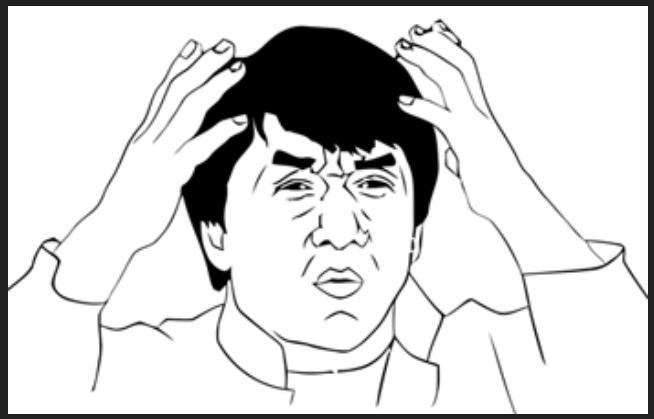
- Storage of Android-Phone disk is limited
  - Extended by SD card
    - Limited disk extension(64GB)
    - Slower (2MB/s – 10 MB/s)
  - Backup in Cloud Drive
    - More Storage
    - Much Slower
  - Backup in PC and other devices
    - Have to connect to a computer

Problem

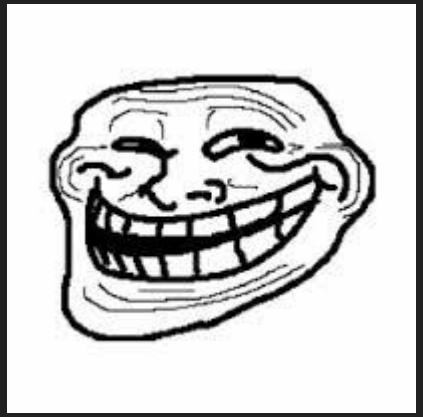
- SD card, Cloud Drive, PC backup
  - Require Manual File management
    - Complex, awful user experience!!!!

Problem

- SD card, Cloud Drive, PC backup
  - Require Manual File management
    - Complex, awful user experience!!!!



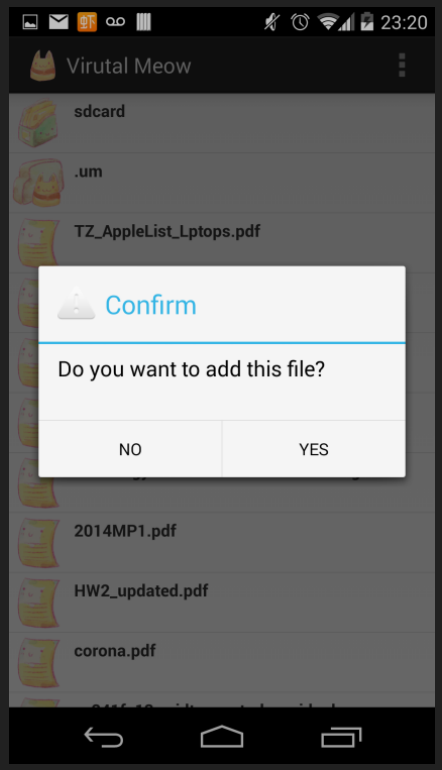
Our Solution: Virtual File System

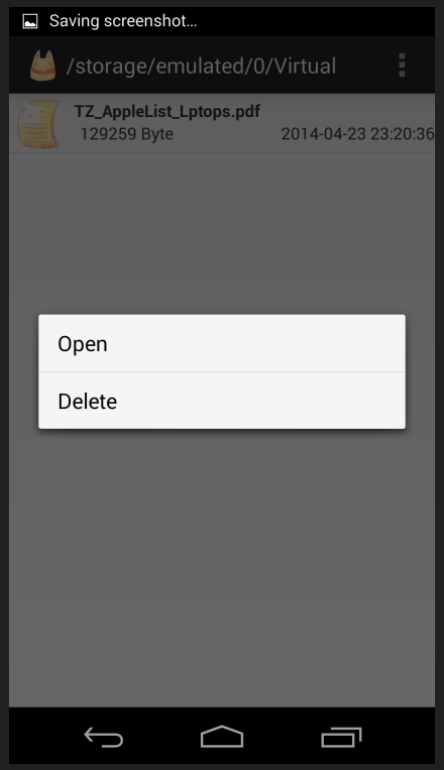


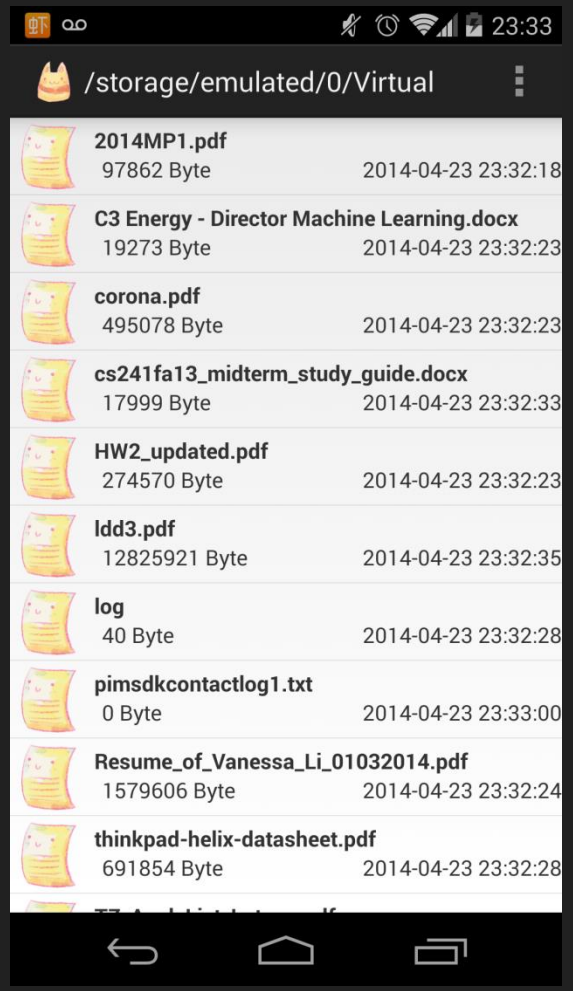
Our Solution: Vitual File System

- Based on Cloud Drive
  - Use limited local disk as a cache
- Transparent File Transfer
  - between local disk and cloud drive
- Automatically File Management
  - No need for root, run in Userspace

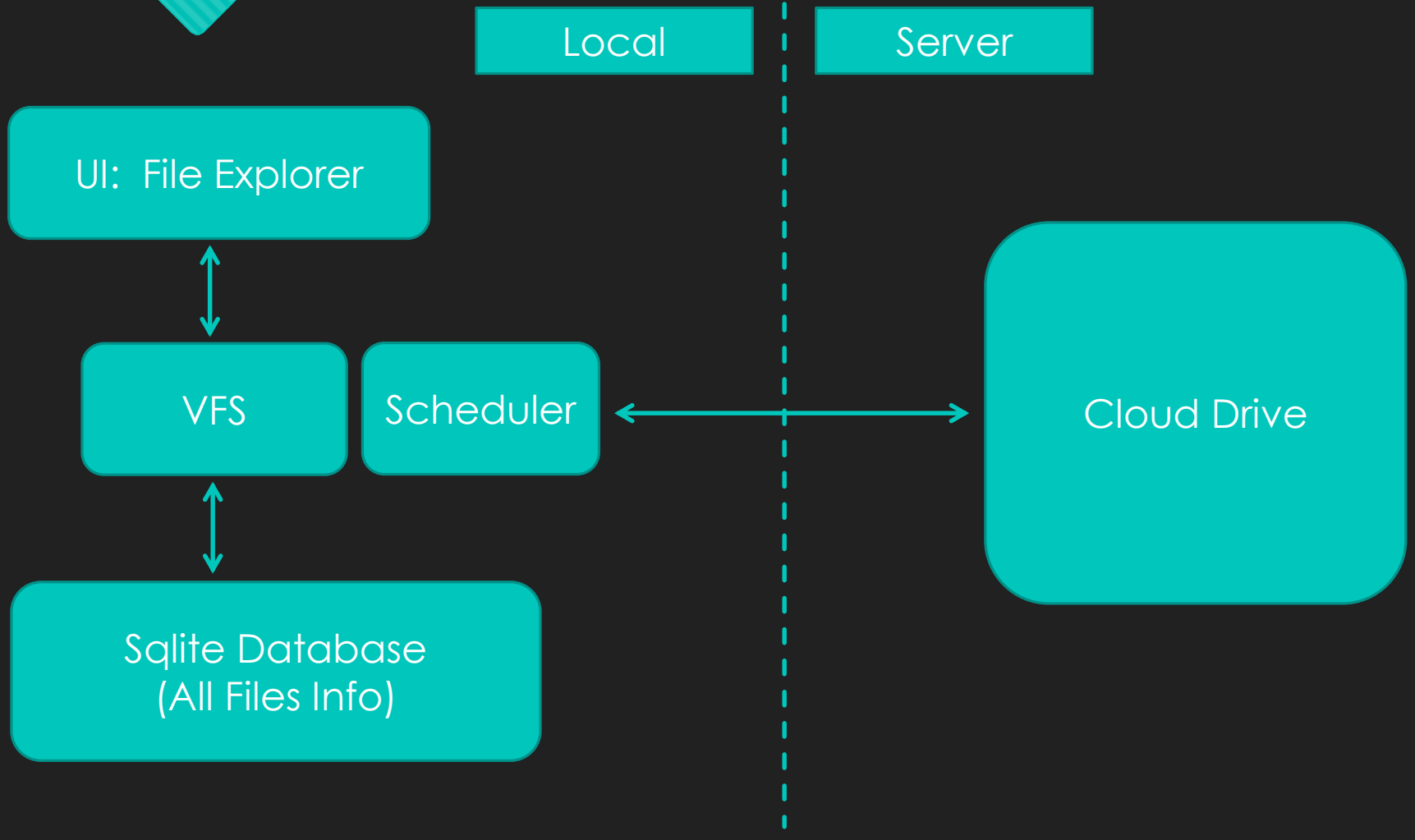
Our Solution: Virtual File System







Architecture



SQLite Database

- Id – each file will have a unique id
- Name – the name of the file
- Path – absolute path of the file
- last_visit_time –time that the file is last visited
- Is_local – is the file stored locally
- Size – the size of the file
- is_directory – the file item a directory
- Probabiilty – used be the GDFS algorithm

Scheduler

- For download, the scheduler will schedule the download immediately.
- For upload, the scheduler will schedule to upload one file per minute.
  - If the file system needs to upload 5 files, the upload task will finish in 5 min.

File Cache Algorithm

- Greed-Dual-Frequency-Size (GDFS)
  - Combination of
    - Greedy-Dual-Size(GDS)
      - Higher file hit ratio, lower bit hit ratio
    - Greedy-Dual-Frequency(GDF)
      - Lower file hit ratio, higher file hit ratio
  - Elegant performance for both file hit ratio and bit hit ratio

GDFS Cache Algorithm

- Priority Queue
  - Priority Value
    - Pr(f) = Clock + Fr(f)/Size(f)
      - Pr(f) : Priority of f
      - Fr(f) : Frequency count of f
      - Size(f) : File size of f
  - Define Used and Total
    - Used as current usage for local storage
    - Total as max usage for local storage

GDFS Cache Algorithm

- Pr(f) = Clock + Fr(f)/Size(f)
  - If request for f is a hit
    - Fr(f) ++
    - Update Pr(f)
  - If request for f is a miss
    - Fr(f) is set to 1
    - Update Pr(f)
    - Used += Size(f)

GDFS Cache Algorithm – cache f

- If Used <= Total
  - Cache f, completed
- If Used > Total
  - Identify the smallest set {f1,f2, …, fk} to evict to satisfy Used - $\sum_{i=0}^{k} Size(fi) \leq Total$

GDFS Cache Algorithm – cache f

- Evict {f1, f2, … fk}
  - If f is not among {f1, f2, …, fk}
    - Set Clock to Max (Pr(fi))
    - Used -= $\sum_{i=0}^{k} Size(fi)$
    - Evict {f1, f2, …, fk} and cache f
  - Otherwise, do not cache f
    - Happens when Pr(f) is too low.
      - Ex. Size(f) is too large
    - Use system default settings to download and open

Future Work

- Support More Storage Extension Solution
  - Support more APIs
    - Commercial Cloud Drive
      - ex. Dropbox, Google Drive
    - Bluetooth Connection to Storage Devices
    - PC and configured server
- File Encryption
- Divide Large File into manageable size

Virtual Meow Demo

Thanks!

- Collaboration
  - Kam Cheung Ting, Tian Xia, Ruogu Zeng
  - {cting4, txia, rzeng4}@illinois.edu

- Reference
  - Cherkasova, Ludmila, and Gianfranco Ciardo. "Role of aging, frequency, and size in web cache replacement policies." *High-Performance Computing and Networking*. Springer Berlin Heidelberg, 2001.

Questions?