ViewBox

Integrating Local File Systems with Cloud Storage Services

Yupu Zhang[+], Chris Dragga[+*],

Andrea Arpaci-Dusseau[+], Remzi Arpaci-Dusseau[+]

[+]University of Wisconsin – Madison

*NetApp, Inc.

Personal Cloud Storage Services

- Exploding in popularity
  - Numerous providers: Dropbox, Google Drive, SkyDrive …
  - Large user base: Dropbox has more than 100 million users

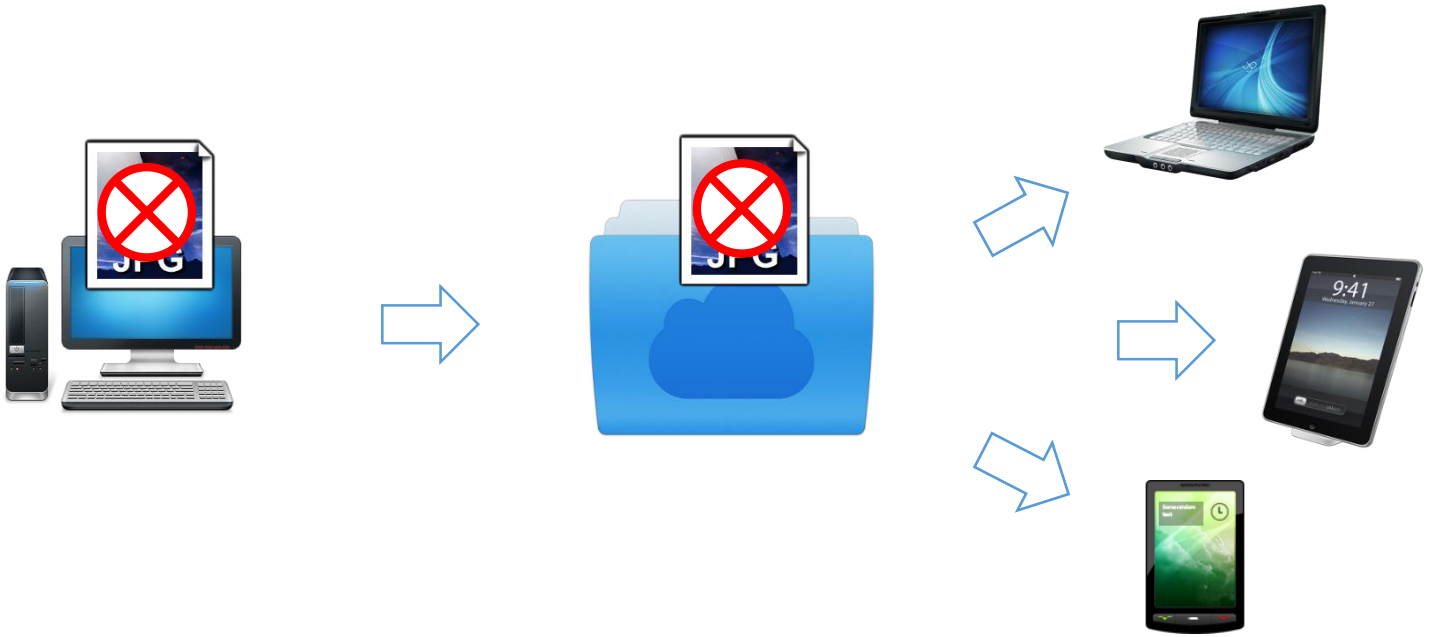
- Promising benefit
  - Reliable backup on the cloud
  - Automatic synchronization across clients/devices

There are so many copies…
My data must be safe...

Really?

Is Your Data Really Safe?

- Data corruption
  - Uploaded from local machine to cloud
  - Propagated to other devices/clients



Is Your Data Really Safe?

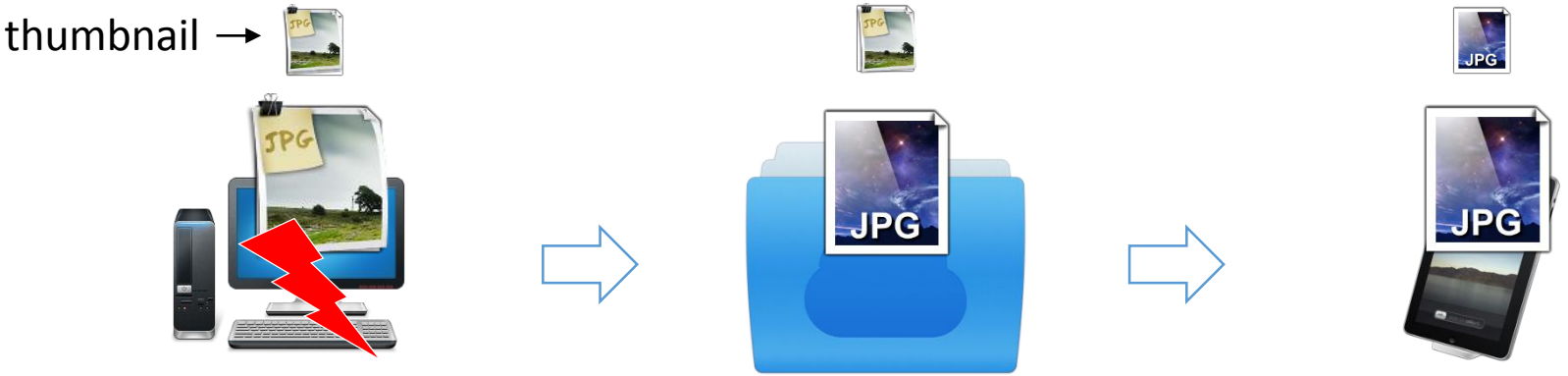
- Crash inconsistency
  - Inconsistent data ends up everywhere
  - "Out-of-sync" synchronization

after reboot

sync client thinks everything is in sync

Is Your Data Really Safe?

- Causal inconsistency
  - Files are uploaded out of order
  - Cloud state does not match a valid FS state



Many copies do
NOT
make your data safe

Why? – File Systems

- Local file system is the weakest link
- Corruption and inconsistency are exposed

file system state ≠ correct state

Why? – Sync Services

- Ad-hoc synchronization is harmful

- Sync client sees what regular application sees, but *not what file system sees*

cloud state ≠ file system state

Can we achieve

cloud state = file system state = correct state

with existing systems?

Our solution: ViewBox

integrated file system and cloud storage

- Local detection + Cloud-aided recovery
  - Rely on strong local file system to detect problems
  - Utilize cloud data to recover from local failures

file system state = correct state

- Orchestrated synchronization based on views
  - In-memory snapshots of valid file system state
  - *Sync client sees what file system sees*

cloud state = file system state

Results

- ViewBox runs on top of existing systems
  - Enhance ext4 with data checksumming
  - Work with unmodified Dropbox and modified Seafile
- ViewBox provides better reliability
  - No global data pollution
  - Automatic recovery with cloud data
- ViewBox incurs minimal overhead
  - Less than 5% overhead for most workloads
  - Up to 30% reduction of synchronization time in some cases

Outline

- Introduction
- Design and Implementation
  - **ViewBox Overview**
  - Implementation
- Evaluation
- Conclusion

ViewBox Overview

- Local detection
  - No corruption/inconsistency is spread

ext4-cksum

- Cloud-aided Recovery
  - Restore file system to correct state upon failure

Cloud Helper

- View-based Synchronization
  - Present file system's view to sync service
  - Basis for consistency and correct recovery

View Manager

Dropbox Architecture

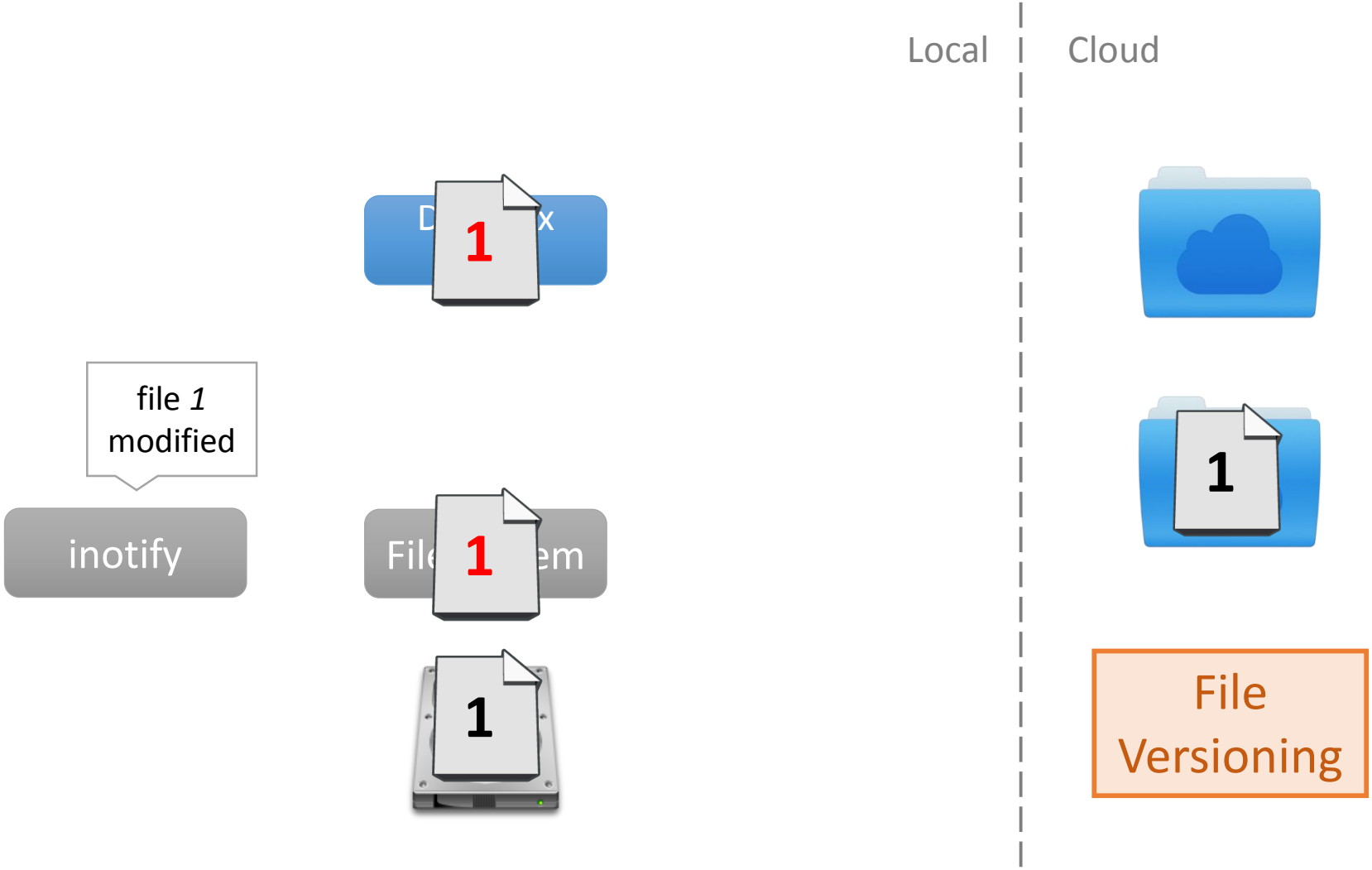
Local | Cloud

Dropbox Client

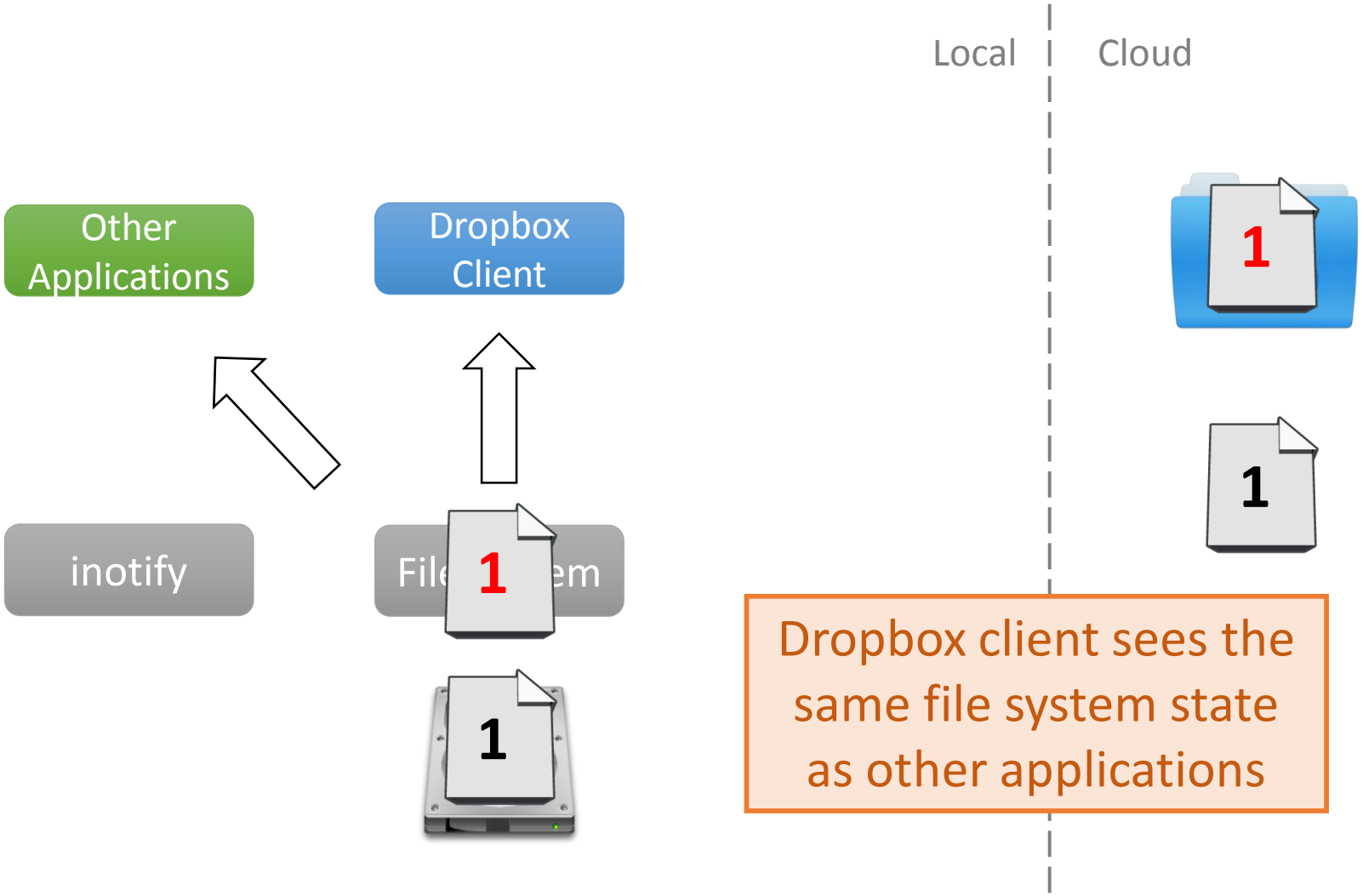
inotify

File System

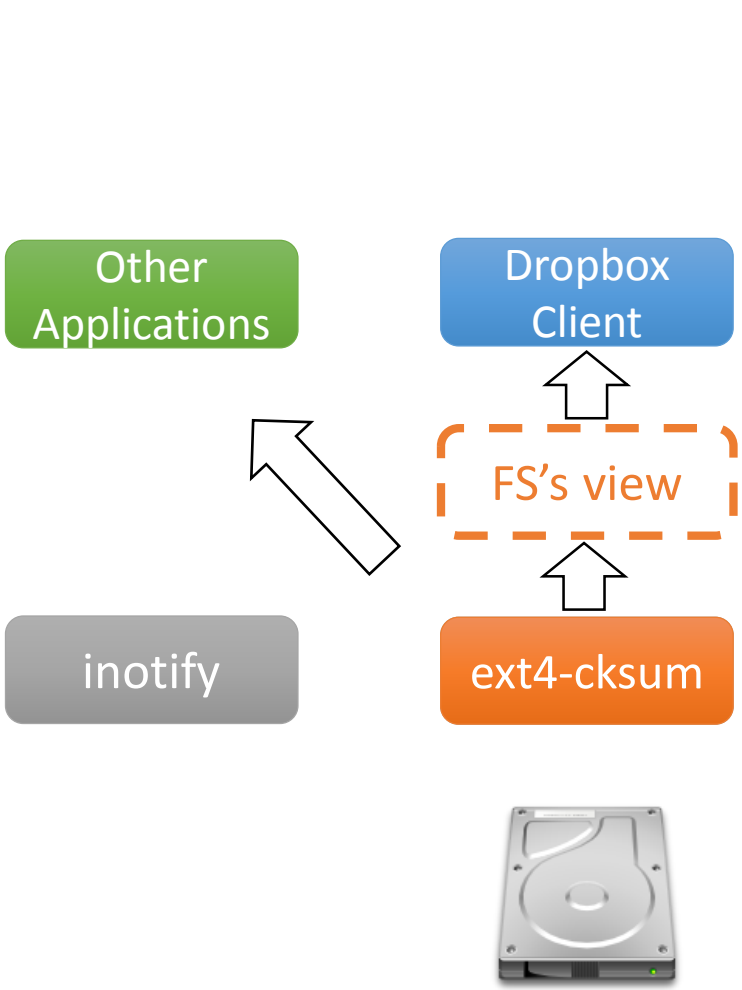
Dropbox Architecture



Dropbox Architecture



ViewBox Architecture



Local | Cloud

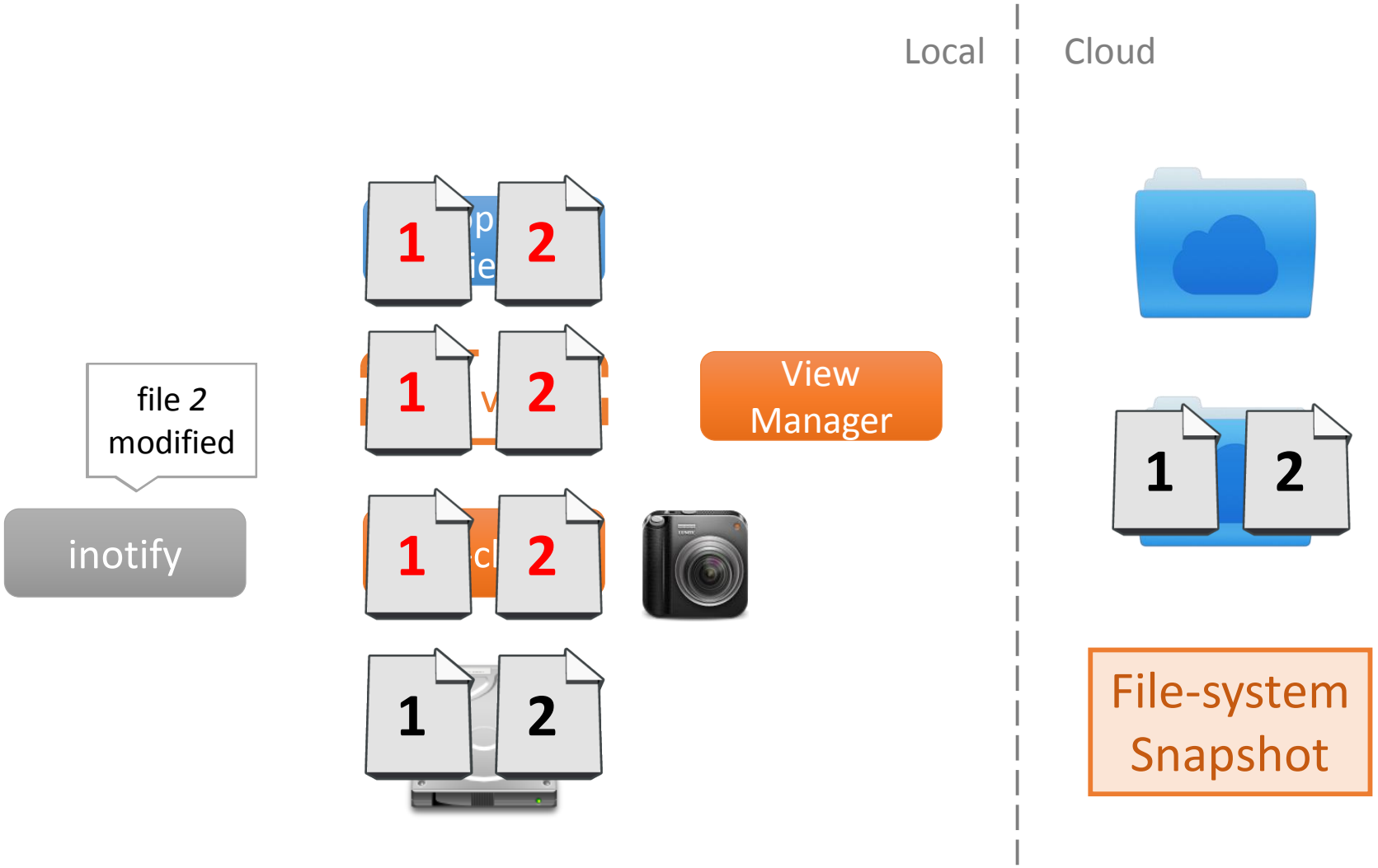
ext4-cksum is dedicated to the entire sync folder

View Manager

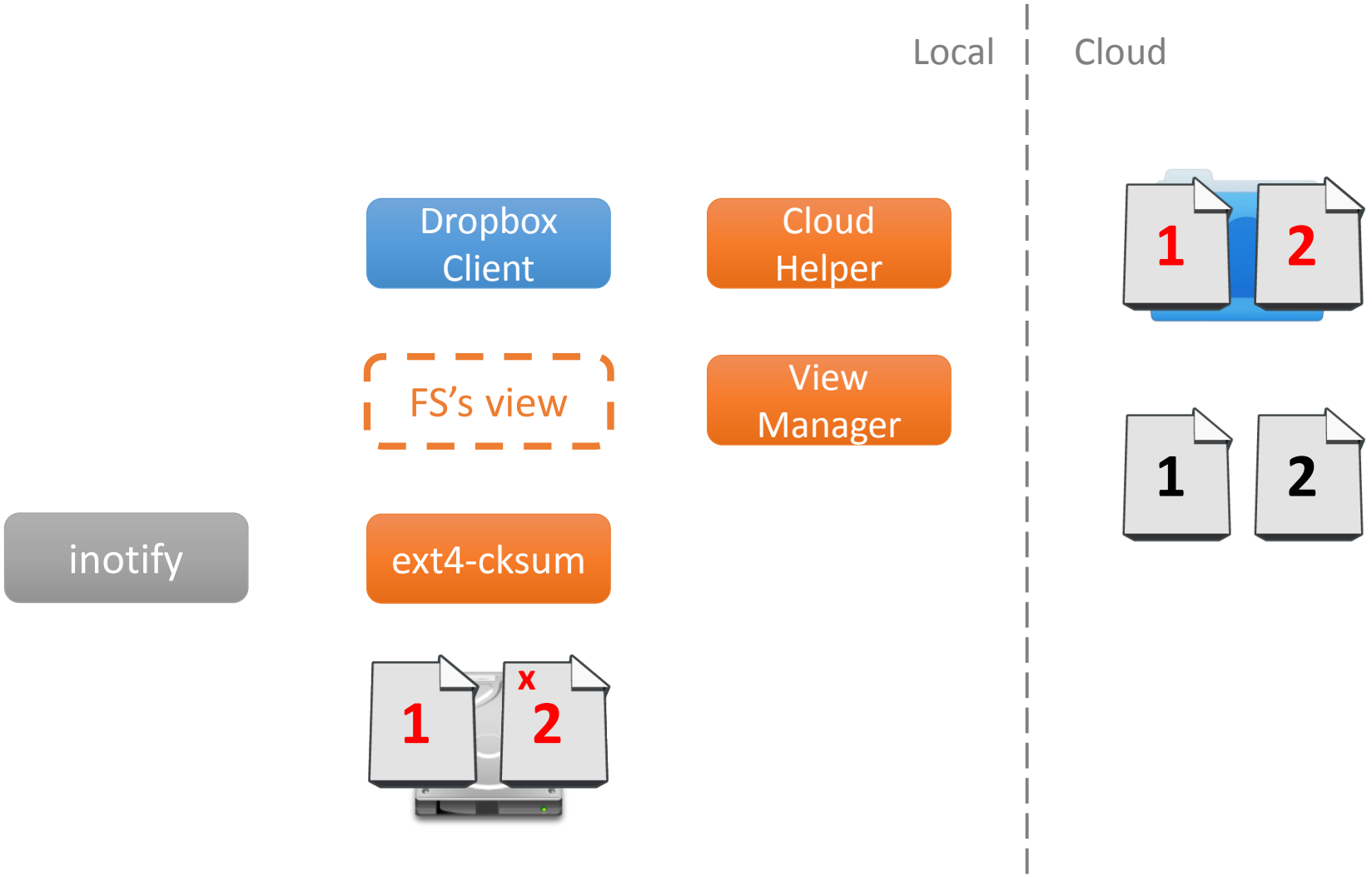
ViewBox presents FS's view to Dropbox client

Other applications' view remains the same

ViewBox Architecture



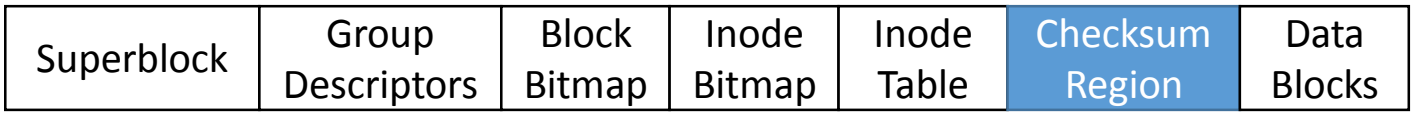
ViewBox Architecture



Outline

- Introduction
- Design and Implementation
  - ViewBox Overview
  - **Implementation**
- Evaluation
- Conclusion

ext4-cksum – Local Detection



- Checksum region
  - Pre-allocated space (~0.1% overhead)
    - 32-bit CRC checksum per 4KB block
    - 128KB checksum region for a 128MB block group
  - Each checksum maps to a data block in the block group
- Detect data corruption & inconsistency

Cloud Helper – Cloud-aided Recovery

- A user-level daemon
  - Talks to local FS through ioctl
  - Communicates with the server through web API
- Upon data corruption
  - Fetches correct block from cloud
- After crash, two types of recovery
  - Recovers inconsistent files
  - Rolls back entire file system to the latest synced view

View Manager – View-based Sync

- Create file system views
- Upload views to cloud through sync client

- Challenge 1 - How to provide consistency?
  - ext4-cksum still runs in ordered mode
  - Cloud journaling

- Challenge 2 - How to create views efficiently?
  - No support from ext4-cksum
  - Incremental snapshotting

Challenge 1: How to Guarantee Consistency?

- Cloud journaling
  - Treat cloud storage as an external journal
  - Synchronize local changes to cloud at FS epochs
    - i.e., when ext4-cksum performs a journal commit

- Three types of views
  - Active view (local) => Current FS state
  - Frozen view (local) => Last FS snapshot in memory
  - Synced views (on cloud) => Previously uploaded views

- Roll back to the latest synced view upon failure

Challenge 2: How to Efficiently Freeze a View?

- Incremental snapshotting
  - Keep previous frozen view in memory
  - Track changes/deltas in active view
  - New frozen view = Previous frozen view + Deltas
- Key to efficiency
  - Decouple namespace and data
  - Directly reflect namespace updates to frozen view
  - Data changes remain in active view but marked COW

Outline

- Introduction
- Motivation
- Design and Implementation
- **Evaluation**
- Conclusion

Evaluation

- Questions to answer
  - Can ViewBox offer integrity, consistency, and recoverability?
  - What is the overhead of ViewBox during user workloads?

- Setup (for both server and client machines)
  - 3.3GHz Intel Quad Core CPU, 16 GB memory
  - 1TB Hitachi hard drive
  - Linux kernel 3.6.11 (64-bit), ~7000 LOC added/modified
  - Dropbox client 1.6.0
  - Seafile client and server 1.8.0

Reliability

- Data Corruption

L: Local corruption G: Global corruption
D: Detected R: Recovered

Service	Data Writes	Metadata Changes		
		mtime	ctime	atime
ViewBox w/Dropbox	D R	D R	D R	D R
ViewBox w/Seafile	D R	D R	D R	D R

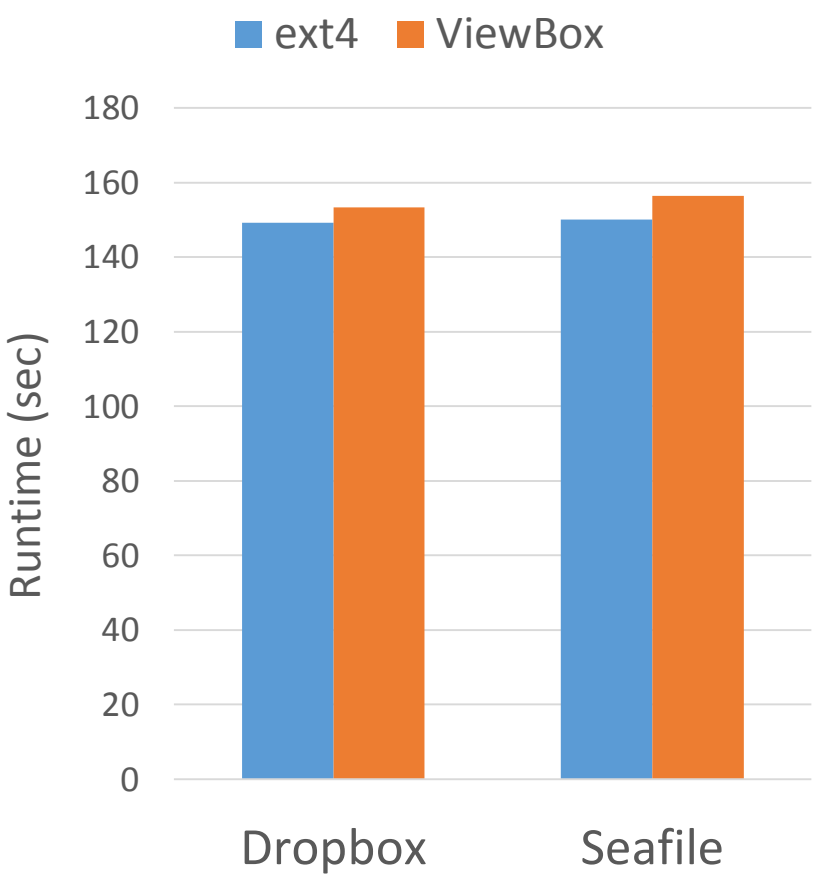
- Crash consistency

YES: occurred NO: did not occur

Service	Upload Local Ver.	Download Cloud Ver.	Out-of-sync (no sync)
ViewBox w/Dropbox	NO	YES	NO
ViewBox w/Seafile	NO	YES	NO

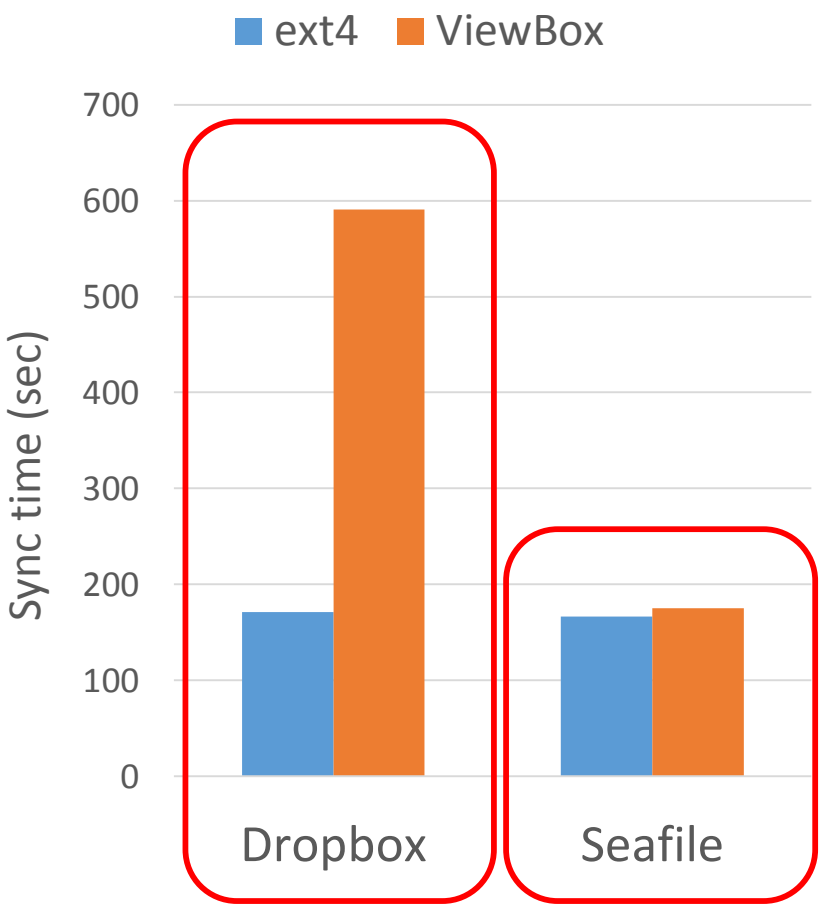
- Causal ordering is preserved

Performance - Photo Viewing



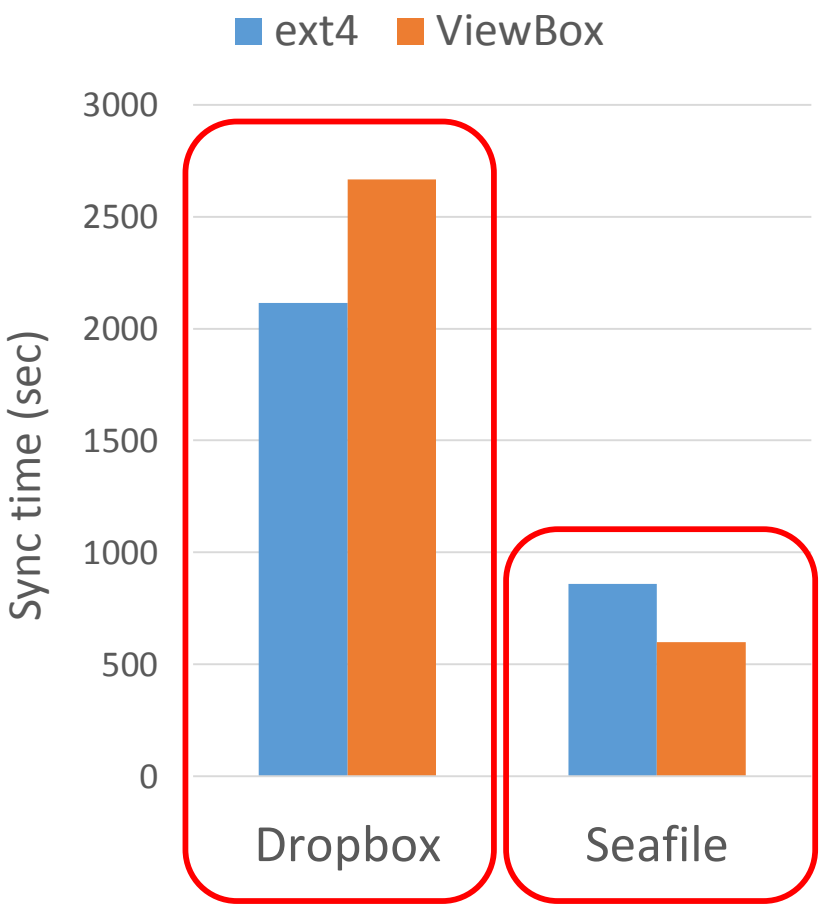
- iphoto_view from iBench [Harter2011]
  - Sequentially view 400 photos
  - Read-dominant
- Runtime
  - Time taken to finish the workload
  - ViewBox has <5% overhead
- Memory overhead
  - < 20MB

Performance - Photo Viewing



- Sync time
  - Time taken to finish synchronizing
- Huge increase in sync time with ViewBox + Dropbox
- View metadata for Dropbox
  - A list of {pathname, version number}
  - Remote walk ~1200 dirs (~1200 RTT) due to lack of proper server support
- View metadata for Seafile
  - Its internal commit ID

Performance - Photo Editing



- iphoto_edit from iBench [Harter2011]
  - Sequentially edit 400 photos
  - Reads:Writes = 7:3
- 30% reduction in sync time with ViewBox + Seafile
- Reduced interference from foreground update
  - Original Seafile may delay uploading
  - ViewBox keeps uploading changes from frozen views

Conclusion

- Problem: Cloud storage services and file systems fail to protect data

  cloud state ≠ file system state ≠ correct state

  - Many copies do NOT always make data safe

- Solution: ViewBox

  cloud state = file system state = correct state

  - Enhance local file systems with data checksumming
  - Present file system's view to sync service

- Tighter integration => more than reliability?

ViewBox: Integrating Local File Systems with Cloud Storage Services

Thanks! Questions?

Advanced Systems Lab (ADSL)
University of Wisconsin-Madison
http://www.cs.wisc.edu/adsl

Wisconsin Institute on Software-defined Datacenters in Madison
http://wisdom.cs.wisc.edu/