ViewBox: Integrating Local File Systems with Cloud Storage Services

Yupu Zhang, Chris Dragga, Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau

WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

WISDOM

Cloud-Based File Synchronization Services

- Incredibly popular: Dropbox has 100 million+ users
- Back up files to the cloud
- Synchronize files across multiple clients

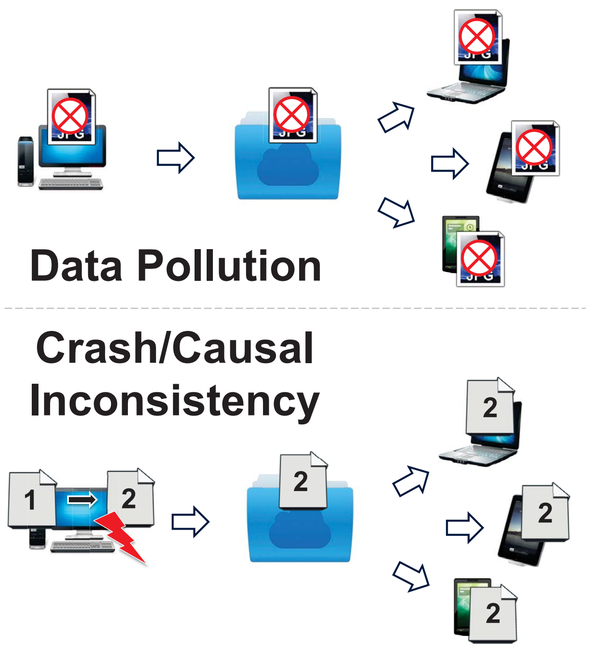
Is Your Data Really Safe?

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

file system state ≠ correct state ✘

- Ad-hoc synchronization is harmful
  - Sync client sees what regular application sees, but *not what file system sees*

cloud state ≠ file system state ✘



Many copies do NOT always make your data safe

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

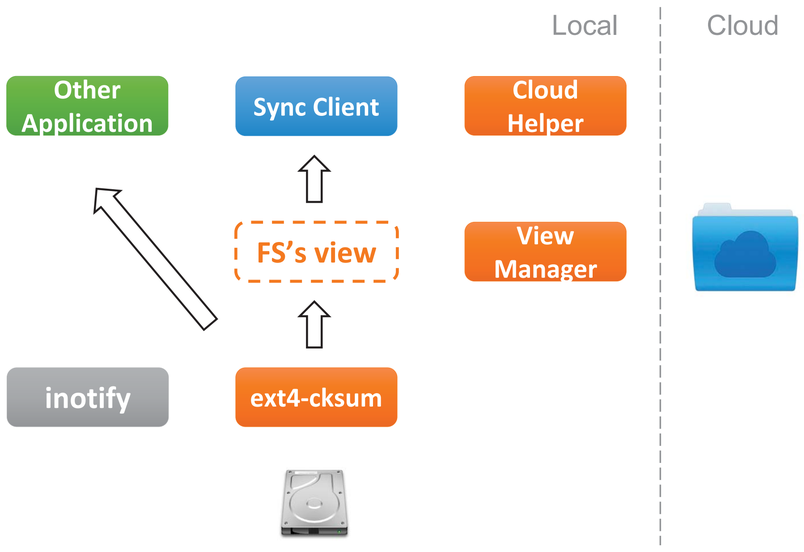
file system state = correct state

- Orchestrated synchronization based on views
  - Views: In-mem snapshots of valid file system state
  - Expose file system's views so that *sync client sees what file system sees*

cloud state = file system state

- Built around ext4 and two services: Dropbox and Seafile
  - Less than 5% overhead for most workloads
  - Up to 30% reduction of sync time in some cases

ViewBox Architecture



- Local detection (ext4-cksum)
  - Detect corruption & inconsistency using cksum
  - Initiate recovery
  - Local FS is dedicated to the entire sync folder
- Cloud-aided recovery (Cloud Helper)
  - Recover from corruption and crashes using synchronized views on cloud
- View-based synchronization (View Manager)
  - Basis for consistency and correct recovery
  - Present file system's view to sync service
  - Other applications' view remains the same
- Three types of views
  - Active view (local) => current FS state
  - Frozen view (local) => last snapshot in memory
  - Synced views (on cloud) => uploaded views

① ext4-cksum

Super block	Group Descriptors	Block Bitmap	Inode Bitmap	Inode Table	Checksum Region	Data Blocks

- Pre-allocated checksum region (~0.1% overhead)
  - Each checksum maps to a data block
  - 32-bit CRC checksum per 4KB block
- Detect data corruption & inconsistency

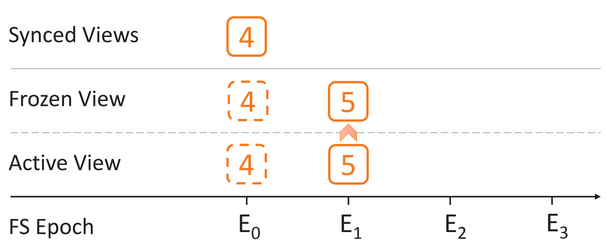
② Cloud Helper

- User-level daemon
  - Talk to local FS through ioctl
  - Communicate with cloud through web API
- Recover from corruption
  - Fetch correct block from cloud
- Recover from crash
  - Recover inconsistent files
  - Rollback FS to latest synced view on cloud

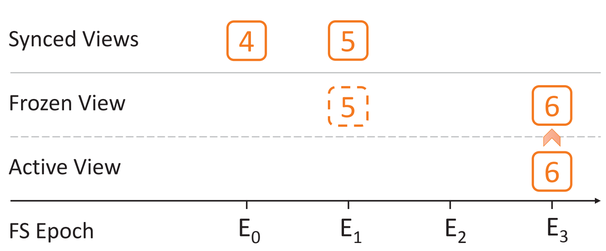
③ View Manager

Tech #1 - Cloud Journaling

Treat cloud storage as external journal



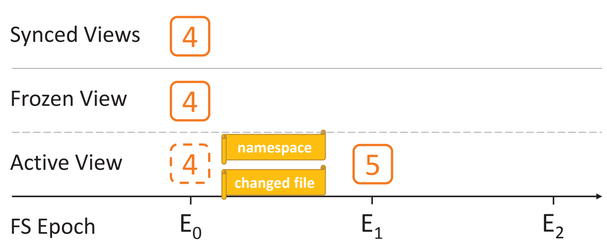
Create frozen views at FS epochs



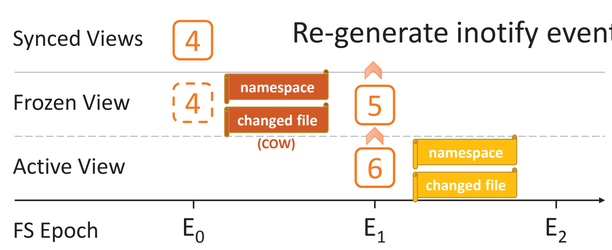
Upload frozen views to cloud
Store multiple views on cloud for recovery

Tech #2 - Incremental Snapshotting

Decouple namespace and data



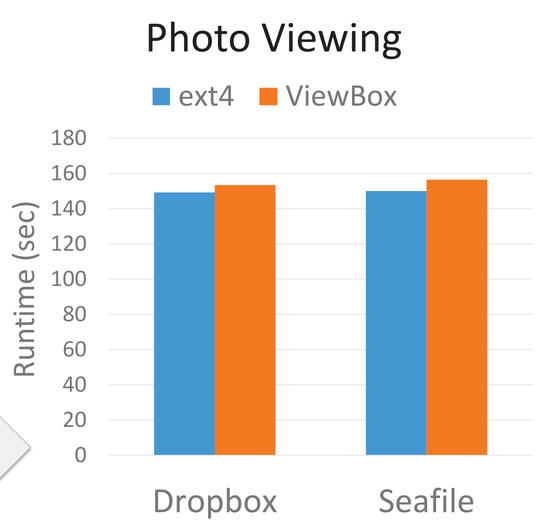
Log namespace changes and data changes



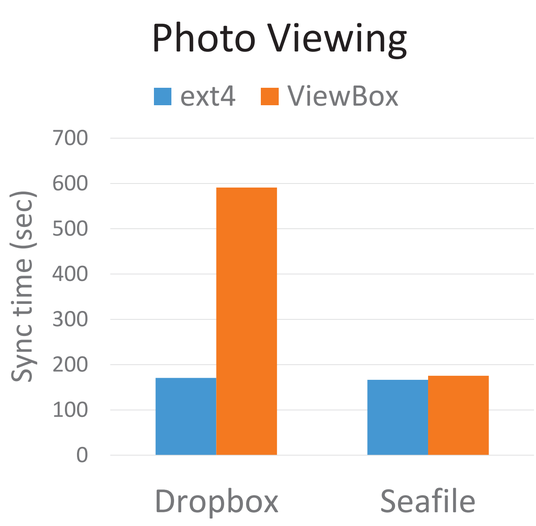
Apply namespace changes to last frozen view
Data is left in FS, but marked COW

Evaluation

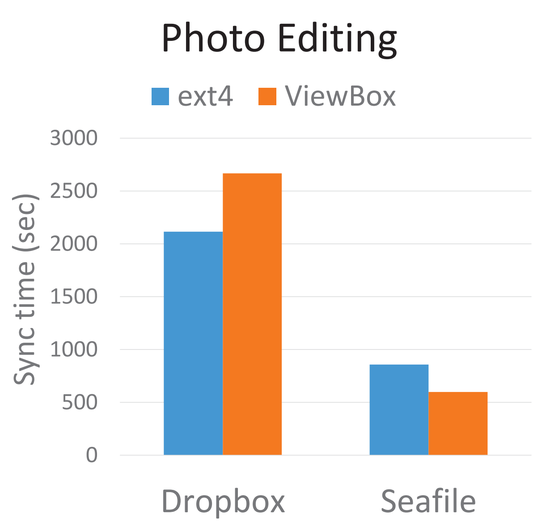
- Reliability => fault injection
  - ViewBox is able to detect corruption and recover from it using cloud data
  - Upon a crash, ViewBox downloads consistent versions of files from cloud
  - Causal ordering is preserved
- Performance => trace replay
  - Replay iphoto_view and iphoto_edit traces from iBench
  - Measure workload runtime and synchronization time



- ViewBox does not affect foreground workload
- Runtime overhead <5%



- Huge increase in sync time with ViewBox and Dropbox, due to Dropbox's lacking of view-API



- ViewBox and Seafile improve sync time due to reduced interference from foreground updates