Project Proposal for Operating Systems Course

Objective:

We plan to design and implement a generic filesystem overlay, capable of providing file-level snapshots, transparent versioning and recovery to any-point-in-time on top of any existing filesystem. Our solution aims to be non-invasive and extensible in nature, possibly at the cost of performance and efficiency.

Scope:

We do not plan on supporting recovery of directories in the initial implementation, primarily to reduce the complexity of implementation. Time permitting we will address this limitation along with efficient storage problem, long term retention policies, transparent compression and encryption feature, data integrity checks, quick filesystem replication and other related features.

Motivation:

Our work is inspired by the lack of an existing production grade filesystem capable of providing a combination of file-level snapshots, transparent versioning and portability.

Wouldn't it be cool to never having to worry about losing any modifications you made to a file? What about guaranteed recovery from 0-day malware attacks?

An Overview:

Existing Approaches (and some comments):

1. ZFS:
- Does not do transparent versioning (though an user-space extension could be built)
- Requires a dedicated filesystem on a block device.
- Has functionality to delete snapshots
- Can’t be natively ported to run on Linux (licensing problems)

2. Dropbox:
+ Supports transparent versioning and centralized backups
+ Works on top of any filesystem and is portable
- Proprietary Solution and requires internet connectivity to work
- The low frequency of snapshot process can lose rapidly changing data (not tested!)

3. HAMMER:
+ Supports full-history retention transparently, data integrity checks
- Requires a dedicated filesystem on a block device.
- No compression support (in general, the design doesn’t seem to emphasize extensibility)

4. Elephant File System:

- Research filesystem, not widely used, porting required in order making it work on modern operating systems.
- Requires a dedicated filesystem on a block device to work.
- Can it handle addition of arbitrary extensions like compression, encryption and data integrity checks without changes to kernel code?

We choose to implement our filesystem overlay using FUSE. This allows our solution to be built and operated in user-space, be non-invasive and extensible. As a design feature, there is no user-level utility to delete versions/snapshots.

Team Member(s):

1. Dhirendra Singh Kholia
2. None