# FILESTORE FILESYSTEM COMPATILIBITY

http://marc.info/?l=ceph-devel&m=131942130322957&w=2

Although running on ext4, xfs, or whatever other non-btrfs you want mostly works, there are a few important remaining issues:

## EXT4 LIMITS TOTAL XATTRS FOR 4KB

This can cause problems in some cases, as Ceph uses xattrs extensively. Most of the time we don’t hit this. We do hit the limit with radosgw pretty easily, though, and may also hit it in exceptional cases where the OSD cluster is very unhealthy.

There is a large xattr patch for ext4 from the Lustre folks that has been floating around for (I think) years. Maybe as interest grows in running Ceph on ext4 this can move upstream.

Previously we were being forgiving about large setxattr failures on ext3, but we found that was leading to corruption in certain cases (because we couldn’t set our internal metadata), so the next release will assert/crash in that case (fail-stop instead of fail-maybe-eventually-corrupt).

XFS does not have an xattr size limit and thus does have this problem.

## OSD JOURNAL REPLAY OF NON-IDEMPOTENT TRANSACTIONS

**Resolved** with full sync but not ideal. See http://tracker.newdream.net/issues/213

On non-btrfs backends, the Ceph OSDs use a write-ahead journal. After restart, the OSD does not know exactly which transactions in the journal may have already been committed to disk, and may reapply a transaction again during replay. For most operations (write, delete, truncate) this is fine.

Some operations, though, are non-idempotent. The simplest example is CLONE, which copies (efficiently, on btrfs) data from one object to another. If the source object is modified, the osd restarts, and then the clone is replayed, the target will get incorrect (newer) data. For example,

- clone A -> B
- modify A
- <osd crash, replay from 1>

B will get new instead of old contents.

(This doesn’t happen on btrfs because the snapshots allow us to replay from a known consistent point in time.)

Possibilities:

- full sync after any non-idempotent operation
- re-evaluate the lower level interface based on needs from higher levels, construct only safe operations, be very careful; brittle
- use xattrs to add sequence numbers to objects:
  - on non-btrfs, we set a xattr on every modified object with the op_seq, the unique sequence number for the transaction.
  - for any (potentially) non-idempotent operation, we fsync() before continuing to the next transaction, to ensure that xattr hits disk.
  - on replay, we skip a transaction if the xattr indicates we already performed this transaction.

  Because every ‘transaction’ only modifies on a single object (file), this ought to work. It’ll make things like clone slow, but let’s face it: they’re already slow on non-btrfs file systems because they actually copy the data (instead of duplicating the extent refs in btrfs). And it should make the full ObjectStore iterface safe, without upper layers having to worry about the kinds and orders of transactions they perform.