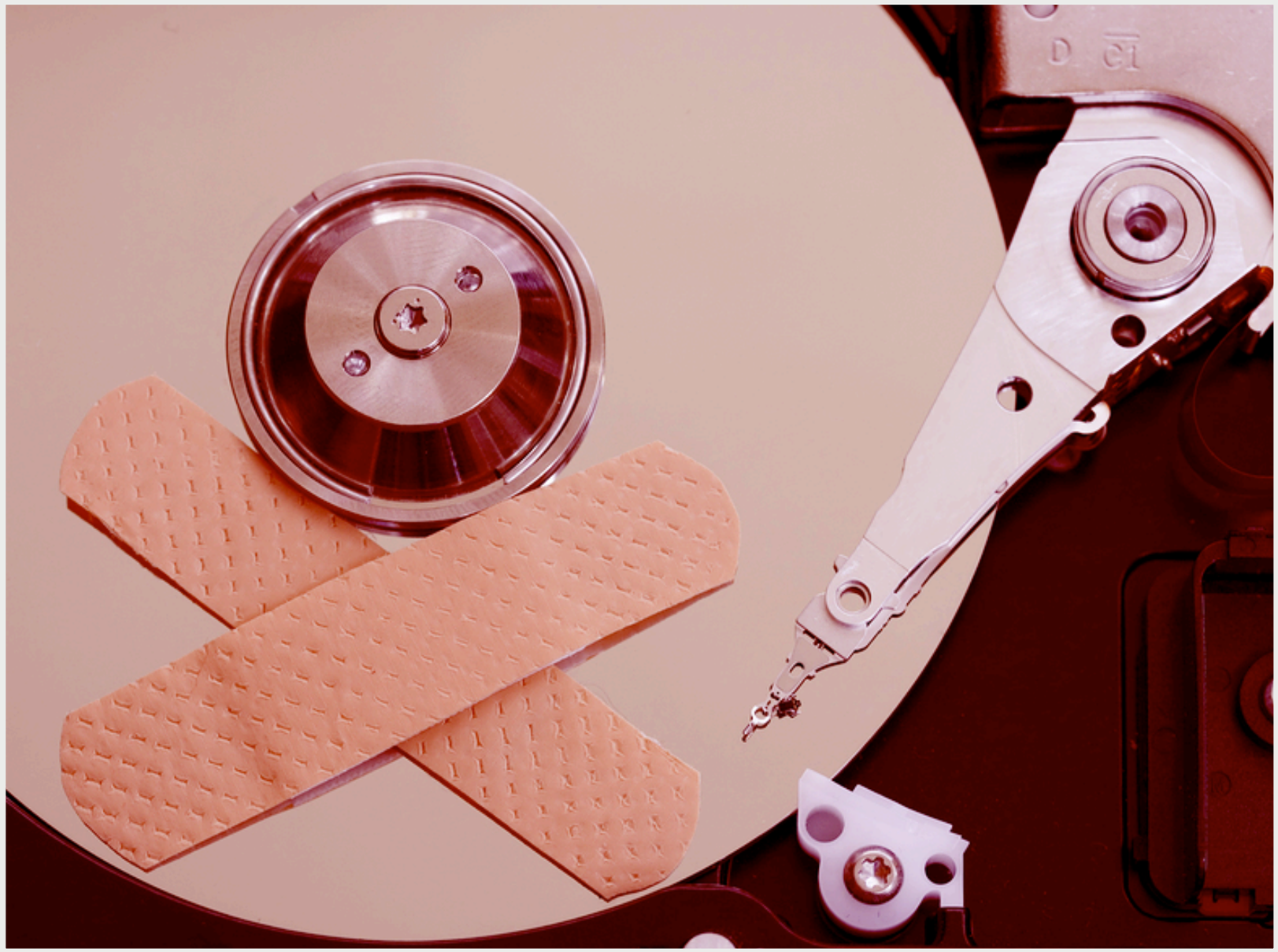
# RECOVERY OVERVIEW

Robert Read



# RECOVERY

- Imports
- Adaptive Timeouts
- Version Based Recovery
- Commit on Share
- Interoperable Recovery
- 2.0 Recovery Changes

# IMPORTS

# IMPORTS & EXPORTS

- An Import is the client side of the connection to a service.
- Export is the server side.
- A client has one import for every service.
- A service has one Export for every client.
- Recovery state is managed by Imports and Exports.

# IMPORT STATUS

- Import status /proc file (1.6.7 and above)
- Dumps current import state
  - cat /proc/fs/lustre/{mdc,mgc,osc}/*/import
  - lctl get_param '*.*.import'

# SAMPLE IMPORT

```
# cat /proc/fs/lustre/mdc/lustre-MDT0000-mdc-cde80000/import
import: lustre-MDT0000-mdc-cde80000
    target: lustre-MDT0000_UUID@10.0.1.180@tcp
    state: FULL
    inflight: 0
    unregistering: 0
    conn_cnt: 37
    generation: 6
    inval_cnt: 0
    last_replay_transno: 1329
    peer_committed_transno: 1338
    last_trasno_checked: 1338
    flags: replayable pingable
```

# ADAPTIVE TIMEOUTS

# ABOUT TIMEOUTS

- Timeouts determine when server is unresponsive
  - Server failure (or failover)
  - Network partition
  - Heavy load/net congestion
- Server death and load appear identical
- Longer timeouts increase recovery time

# WHEN TIMEOUT OCCURS

- In-flight RPCs are saved on resend list
- New RPCs are delayed
- Client attempts to reconnect to the server (or failover) until successful
- Client resends in-flight RPCs and sends are delayed

# ADAPTIVE TIMEOUTS

- Two Goals:
  - Automatically adjust RPC timeouts as network conditions and server load change
  - Decrease server recovery time
- Differentiate between dead server and busy server.

# NORMAL OPERATION

- Timeout set per RPC based on current info
  - server response time
  - network lag
- Server sends early replies to increase in-flight RPC timeouts
  - prevent reconnect/resend

# SERVER RECOVERY

- Initial recovery period based on obd_timeout
- As client reconnect,  server adapts based on previous server response times
  - active clients have similar timeouts
  - need to wait until all clients timeout their RPCs

# VERSION BASED RECOVERY

# PROBLEMS WITH RECOVERY

- Strict in-order replay requirement (no gaps)
- Requires all clients to reconnect during recovery period
- All clients evicted if recovery times out

# SOLUTION

- Uses versions to detect conflicting transactions
- If version on an object is what we expect, then we can replay transaction
- Object version mismatches is a "mini-gap"
  - Only stops the clients attempting to modify that object
- Transactions on other objects can continue

# NEW RECOVERY

- Conditional out-of-order replay is allowed
- There is a still a limited recovery period
- Clients blocked by gaps are evicted at end of recovery
- Clients that have not reconnected are not evicted

# DELAYED CLIENTS

- A client that reconnects after server recovery ends
- Transactions are replayed with same version matching rules
- Eventually state for delayed clients is purged

# COMMIT ON SHARE

# DETECT CONFLICTS

- Checks for an uncommitted transactions from a different client before updating an object.
- Commit first, then update
- All uncommitted transactions have no dependencies

# INTEROPERABLE RECOVERY

# INTEROP RECOVERY

- Use case: Failover upgrade
  - upgrading from Lustre 1.8 to 2.0
- RPC format has changed in 2.0
- Recovery in 2.0 is different
- Original plan was for clients to reformat requests

# SIMPLIFIED INTEROP

- Server and clients coordinate before shutdown
- Minimize state that needs to be replay
- Still uses failover / recovery mechanism to resume

# MANAGED FAILOVER

- Clients are notified of pending upgrade
  - Flush all dirty state and release locks
  - Wait until data is committed
  - Only left to replay are open files
- Server stops cleanly

# 2.0 RECOVERY CHANGES

# OST GROUPS

- notion of a "group" has been added
- Support multiple MDSs connected to each OST
- MDS is assigned a group
- pre-creation and orphan cleanup is within a group

# FIDS

- Client generates the fid
- simplifies recovery
- don't send ino/generation  with replay
- no need to reuse the same ino during replay

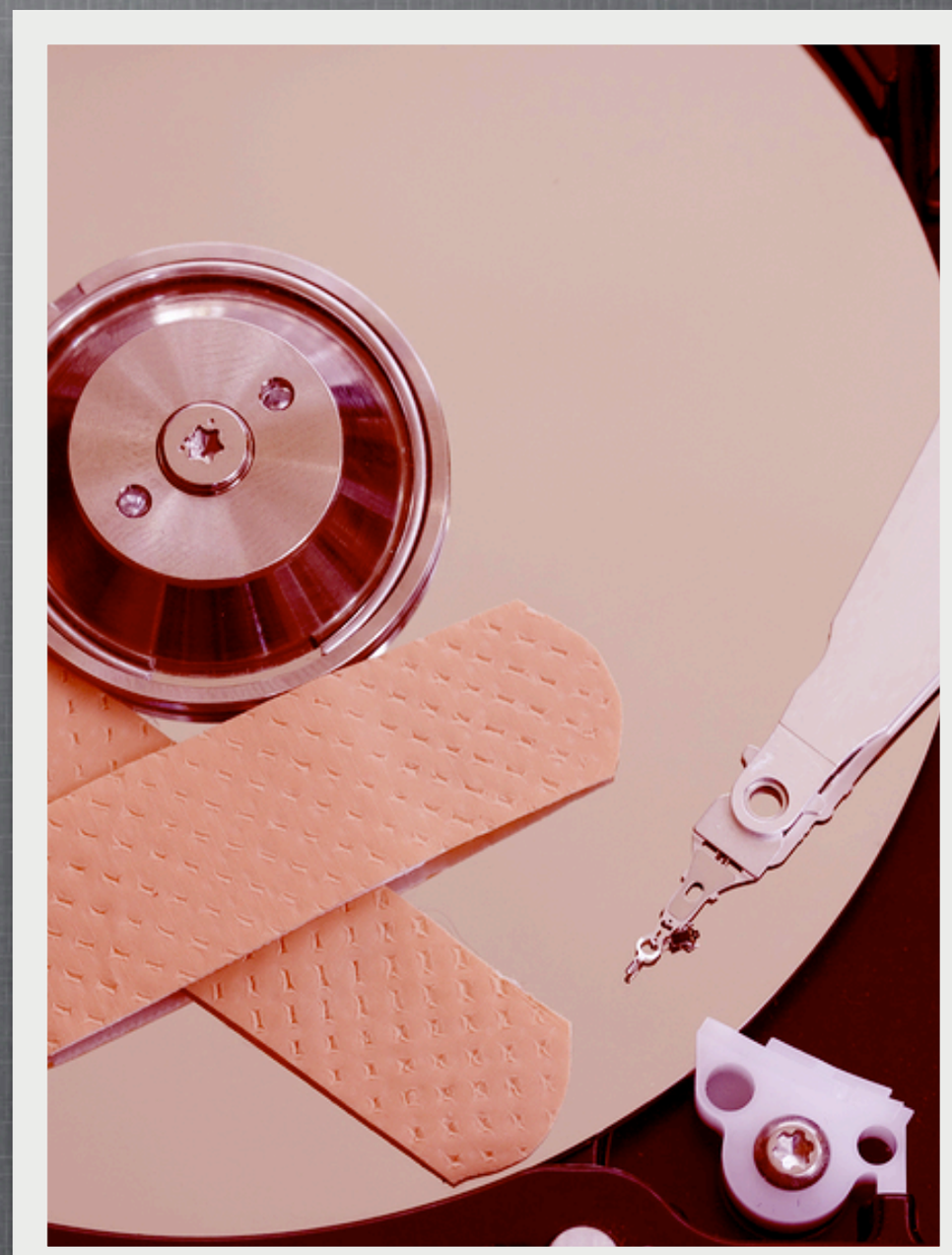
# CMD RECOVERY

- multi-MDS operations not supported
- eg. need mechanism to recover global on-disk consistency after total power-off.
- currently being discussed

# IMPERATIVE RECOVERY

- Currently just an proposal
- Notify all clients server has failed and the new nid
- Server can set recovery period to be much smaller

# RECOVERY OVERVIEW



Robert Read
rread@sun.com