NFSv4 Client Recovery: Good Idea or Disaster Waiting to Happen?

Mike Kupfer
mike.kupfer@sun.com

Payoff versus Danger

- Payoff: shared state lets us do things like delegation
- Danger: increased complexity could result in unreliable implementations

Complexity

- 38 operations × 13 errors that might require recovery
  - not all combinations need be considered
    - of course, that's more complexity
- cascading recovery
  - example: timeout or `STALE_CLIENTID` while recovering open files

Reliability

- Client recovery is unavoidable
- Client recovery will be used when things are already screwed up

It's got to work

Mitigation

- Design
- Logging, audit trail
- Testing

Design

- Decouple ops from error
- Decouple ops from recovery code
- No ops while doing recovery
- Recovery code must be able to restart, handle cascading errors
  - keep ordered list of items that need to be recovered
  - error → update list, start over from the top

Logging

- Server, filesystem, reason for recovery
- Individual files not logged: too much
- Errors

Testing

- internal alpha testing
- force recovery
  - force reboots during stress test
  - force random reboots? (slow)
  - some sort of fault injection? (less impact on users, but could require extensive server hooks)
- should be tested at bakeoff(s)