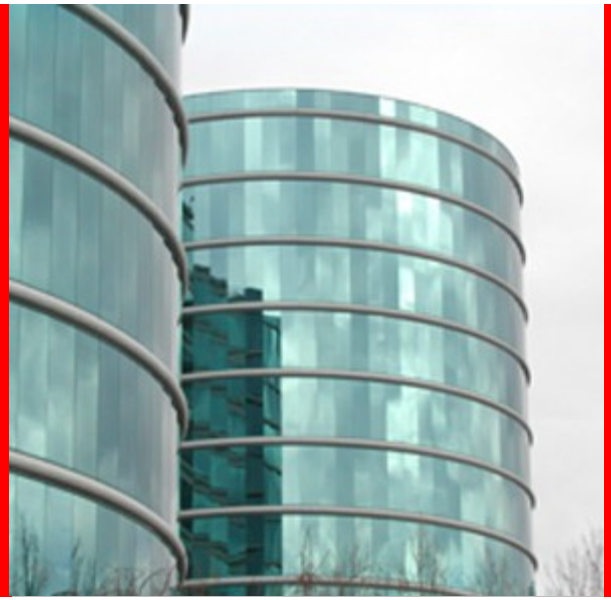

Sun

ORACLE

ORACLE THIRTY YEARS 30

Solaris NFSv4.1 Client – Lessons Learned

Bill.Baker_at_oracle.com

Overview

- Implementation experiences NFSv4.1
- Highlight flaws in our thought process
- Conclusions

History

- Original implementation was a prototype for pNFS client
- False assumption: 4.1 code base was stable and complete

Number of “stateful” compounds has changed

- NFSv4.0
  - Only open, close, lock, locku, etc
  - State recovery not needed for others
- NFSv4.1
  - Most compounds are sequenced
  - These compounds need BADSESSION recovery
  - Yes, all of them. No, not an optional “feature”.
- Legacy code: We don't need no stinkin' recovery
  - Existing code tricked out of the recovery system
  - Failed to use existing interfaces properly

False Sense of Correctness

- Lame, “happy path” functional testing
  - I.E. connectathon
- Client needs to recover from BADSESSION on any compound
- Massive restructuring needed in places
  - Especially when functions make multiple OTW calls
- Testing painfully manual, hard reboots from kmdb

BIG rule: slow sleeps

- If the code does “slow” (indefinite) sleeps, then it must be interruptible
  - `cv_wait_sig()` in RPC
- We are STILL sending RPCs from the user's thread
- ^C out of sequenced operations is bad news
- 4.0 had same problem, but fewer stateful ops

Lesson Learned

- Never trust the user's thread
- Send stateful RPCs via queued, async RPC
  - Sadly, we still aren't doing this
  - Beatings will continue until morale improves

Dealing with signals

- LORP
  - LOst Request Processor
- Use protocol replay mechanism
  - For 4.0, openowner & lockowner replays
- Determine the result of the original operation
  - May require the operation to be “undone”
  - E.g. lock, ^C, resend-lock, OK, unlock, OK

Dealing with signals

- NFSv4.1 raises the stakes
  - Most compounds are sequenced
  - All sequenced operations need this protection
- Use slot replay mechanism
  - Deal with UNCACHED, FALSE_RETRY, OK
  - Discard unneeded replay, app lost interest
- Multiple prototypes tested
  - State recovery engine, bulky
  - Async thread, complicated
  - Lazy recovery of slot via singleton sequence

Dealing with signals ...

- Interruptibility also applies to mount/unmount
- Solaris VFS layer not friendly to errors
  - We had to change the behavior of VFS_MOUNT
- New mount might have to participate in recovery
  - BADSESSION, but still needs to be interruptible
- Go fully async to allow user thread to interrupt out

Dealing with signals ...

- Helen loves to hit ^C
  - Don't we all, but
  - Helen doesn't scale
  - Let's amplify Helen by 100 fold
- New testing methodology – mumo
  - Mount/umount, but generate continuous signals
  - Extended to apply to any workload
  - Shake-n-break + fuzz
  - Combine with server reboot? OMG!
- Very powerful, very painful

Conclusions & Summary

- Question your assumptions as requirements shift
- Testing is not proof of correctness
- **Testing is not proof of correctness!**
- Never send stateful RPC in the context of user thread
- Async threads ratchet up complexity, use with care


ORACLE®