Duplicat Request Cache (DRC) for NFSv4 over TCP
Rick Macklem, University of Guelph

Chet’s Good Ole DRC for NFS over UDP

- **Typically a Global LRU Cache**
- **Since UDP retries agressively**

→ **throw away Hits while "in progress"**
→ **cache replies for non-idempotent RPCs**
→ **reply from cache for Hits with cached replies**

- **some concern w.r.t. False Hits (reused xid values)**

DRC for NFSv4 over TCP

- **only retries after reconnect**

→ never on same TCP connection

- **1 minute to hours later**

→ other clients could still be active

→ global LRU doesn’t work well

- **what about LRU per TCP connection?**
- **time based cache invalidation?**
- **size based cache invalidation?**

☞ **Ideally, when the client has processed the reply**

→ **I invalidate when client side TCP acks receipt of reply (at least reply is in client rcv Q)**

- **Wired down entries for Seqid# Ops**

False Hits

- **all RPCs same # (Compound)**
- **time cached much longer**

→ risk of False Hit greater

‡ Therefore:

I assume a hit "in progress" → False Hit
allow multiple requests for same <cache key>

- **use checksum on first bytes of NFS XDR**

☞ **I didn’t use client IP# in cache key**
(DHCP lease expiry → different IP#)

When request arrives, match all:

if
same xid
‡ not "in progress"
not same socket
same length of NFS XDR
same checksum for first ≤ 100 bytes of XDR
1 hit with no Seqid# Op in it
→ reply with cached entry

During processing of Compound:

```
if non-idempotent Op → set flag
if Op uses a Seqid#
    if same seqid# as referenced entry → Hit
        free this entry
        if cached entry "in progress"
            drop request
        else
            reply from cached entry
    else if next seqid# in order
        free referenced cache entry
        wire down this cache entry
    else if first seqid#
        wire down this cache entry
```

End of Compound Processing:
if wired down OR (non-idempotent AND below Floodlevel)
save reply in cache entry
timestamp it
note TCP seq#
else
free cache entry
Send reply
- **Certain error replies aren't cached:**
  NFS4ERR_GARBAGE, NFS4ERR_BADXDR,
  NFS4ERR_BADSEQID, NFS4ERR_RESOURCE,
  NFS4ERR_STALECLIENTID,
  NFS4ERR_OLDSTATEID, NFS4ERR_BADSTATEID,
  NFS4ERR_GRACE, NFS4ERR_NOGRACE,
  NFS4ERR_MOVED, NFS4ERR_STALESTATEID,
  NFS4ERR_SERVERFAULT,
  NFS4ERR_DELAY - Unless a Seqid# Op

Cache Invalidation Happens When:

- **For Seqid# Op RPCs → next Seqid# Op processed**
- **Others → client TCP acks receipt of reply**
  OR → large timeout (12 hours)

NFSv2 and 3

- **Over UDP → same old DRC**
- **Over TCP, Not the same as NFSv4**
  - **no check for different TCP socket**
  - **cache key includes RPC#**

Although my current code doesn't do so, I think:
‡ should drop request and TCP connection

Packrats: A Work starting to Progress

purely experimental, to see if agressive client side caching will improve perf over network interconnects will large "bandwidth * delay"

packrat threads do agressive client side data caching onto local storage when the server issues a Delegation to the client (whole file copies to the client as soon as the delegation is issued)

delegations are working, but the packrat threads aren’t yet