Reflections On Client Instance Uniqueness

Chuck Lever
Consulting Member of Technical Staff
Constructing `nfs_client_id4`

- What is a client instance?
- RFC 3530bis recommendations
- Existing implementation practices
- Better choices
What Is A Client Instance?

- **Boot verifier - nfs_client_id4.verifier**
  - Unstructured 8-byte value
  - Changes across client reboot
  - Allows server to distinguish between client reboot and callback update

- **Client string - nfs_client_id4.id**
  - Opaque array of bytes chosen by client to be unique from all other clients
  - Fixed across client reboot

- **Principal used for SETCLIENTID operation**
  - Authentication flavor plus credential
  - Gray area
RFC 3530bis

- Section 9.1.1 covers client identification
- When boot verifier changes, server cancels client’s leased state
- Id string
  - Unique across all clients
  - Fixed across client reboots
  - Different for each server address that client accesses
  - Don’t assume client’s address is fixed
- Security measure
  - Server can’t cancel lease of a subsequent SETCLIENTID with same ID and new verifier uses a different principal
RFC 3530bis

• Founder’s intent:
  • A client changes the boot verifier only when it reboots
  • Each distinct client has one and only one id string
  • A client always uses the same principal when sending SETCLIENTID
RFC 3530bis

- Recommended contents of id string
  - Server’s network address
  - Client’s network address
  - Possibly a UUID
  - Client host’s serial number
  - A MAC address
  - A fixed timestamp
  - A true random number

- How has this worked out for us?
RFC 3530 Recommendations

Network Addresses

• Client network address in \textit{id} string
  • Clients can share same address if behind a NAT router
  • Dynamically assigned client address can change over a reboot
  • Multi-homed client would generate a distinct lease for each of its network addresses

• Server network address in \textit{id} string
  • Multi-homed server would create separate leases each server address through which the client accesses it
  • UCS clients must use the same string for all servers
RFC 3530 Recommendations

Authentication

• Server MUST NOT cancel lease if SETCLIENTID principal doesn’t match original SETCLIENTID
  • NFS4ERR_CLID_INUSE is returned
  • Does this make principal part of client’s identity?
• Authentication flavor in \texttt{id} string
  • Client generates separate lease for each flavor used
  • Linux added flavor name to \texttt{id} string until recently
• Using AUTH_SYS with SETCLIENTID
  • Most clients use \{ UID 0, GID 0 \}
  • Server not likely to catch reuse by other clients
  • Server could examine machine name part of credential, but still no guarantee of uniqueness
RFC 3530 Recommendations

Hardware Serial Number

• MAC address in `id` string
  • Client with multiple NICs used serially (e.g. wifi, wired)
  • Client with multiple NICs used concurrently (e.g. multi-homed)
  • OS initialization order of NICs is indeterminant
  • Virtualized clients may get re-used MAC addresses

• Machine serial number
  • Aside from privacy concerns...
  • Most hardware platforms do not have a unique hardware identifier
RFC 3530 Recommendations

Additional Uniqueness

• UUID in id string
  • A Type 1 UUID is a MAC address and a time stamp
  • A random-variant UUID would have to be stored somewhere on the client, but is suitably unique

• Client hostname in id string
  • Nothing stops administrators from assigning same hostname
    • Especially challenging if non-FQDN are chosen
  • Hostnames often dynamically assigned, thus not fixed across reboots
  • Usually client hostname is “good enough,” until it isn’t
Good Practices

• Some servers use source address of SETCLIENTID or SETCLIENTID_CONFIRM to detect distinct clients using the same id string
  • Servers should use only the arguments of these operations, not transport addresses
  • Linux server is known to do this, fixed recently
Good Practices

- Some servers use `id` strings to keep leases sorted after a cluster take over
  - Allows an orderly give back, but
  - This design relies on clients using a distinct string for each server they access
  - Not compatible with UCS approach, needed to support Transparent State Migration
  - SunStorage is known to do this
Good Practices

• Some clients use a unique id string for each mount point
  • Servers must maintain more leases
  • Client must store these strings permanently to permit proper state recovery
  • Intent of RFC 3530 was for each client to use the same id string for all of its mount points
  • FreeBSD is known to do this
Good Practices

- Updating `nfs_client_id4.verifier`
  - RFC 3530 does not require a client’s boot verifier to remain unchanged during a single client restart
  - Servers generally do not check for a verifier replay
  - Server uses boot verifier to prevent loss of leased state during a callback update
  - Some clients change the boot verifier on the next mount after the last mount of a server is gone
  - A client can use a boot verifier change to force a server to remove its leased state
  - Linux client is known to do these last two
Good Practices

• Virtualization

  • Virtualized NFSv4 clients running on the same physical host should use distinct id strings and boot verifiers

  • Virtualized NFSv4 servers running on the same physical host should behave as independent server instances
    • Maintain separate clientid4 spaces
    • Maintain separate leases for a particular client
An Example: Linux Client

• Non-uniform
  • Traditional, now default for NFSv4.0
  • Allows compatibility with existing NFSv4.0 servers
  • Server IP address, client IP address, callback netid

• Uniform
  • Default for NFSv4.1, allowed for NFSv4.0 with migration
  • “Linux,” NFS version, then a uniquifier
    • uniquifier is normally client’s nodename
    • A replacement can be specified on boot command line
      • Can be stored by GRUB or provided via DHCP boot
      • Might be a UUID generated during client installation
An Example: Linux Client

- Traditional boot verifier
  - 64-bit time stamp stored in per-lease data structure
  - Regenerated on next mount when last mount of a server goes away

- New boot verifier
  - 64-bit time stamp stored per container
  - Regenerated when container is created (typically once per boot)
  - Special NFSv4.1 behavior when a STATE_REVOKED sequence flag is asserted
    - EXCHANGE_ID presents an impossible time stamp
    - Second EXCHANGE_ID presents original verifier