NFSv4.1 dynamic slot allocation

Trond Myklebust
<trond@netapp.com>
What is dynamic slot allocation?

- A tool for managing **global** session resources
  - Allows dynamic resizing of the replay cache on a per-client, per-load basis
  - The client communicates to the server whether or not it can fill all slots.
  - The server then decides how many slots it should allocate to that client in the future.
- Communication occurs via the SEQUENCE operation, which means that updates occur on every COMPOUND.
Ordinary session management

- Number of session slots negotiated at CREATE_SESSION time
  - `ca_maxrequests` sets the table size
  - Server pins `sr_highest_slotid` and `sr_target_highest_slotid` to `ca_maxrequests-1`
  - Server ignores the client settings of `sa_highest_slotid`

- If the server runs out of resources, it can force renegotiation of the session by returning `NFS4ERR_BADSESSION`.
Dynamic session management

- Initial session table size still negotiated at CREATE_SESSION time.
  - Session table size changes communicated using SEQUENCE: \textit{sr\_highest\_slotid} and \textit{sr\_target\_highest\_slotid} reply fields
  - Server may adapt table size using its own policy criteria. E.g. client load, resource availability
  - Also a callback mechanism for out-of-band slot recalls.
How does the client communicate load?

- The session slots are numbered from 0…n.
- The client is required to allocate all slots from 0…n-1, before it can use slot n.
- In each SEQUENCE call, the client fills the `sa_highest_slotid` field to reflect the highest slot number in use at the time the SEQUENCE was sent.
How does the server reply?

- The server fills the `sr_highest_slotid` with the highest slotid that the client is allowed to use.
  - This is the highest slotid for which the server is caching the sequence number.
- It fills the `sr_target_highest_slotid` with the highest slotid that the client should use in the future.
  - IOW: as soon as the client sees this target, it should stop allocating new slotids > target.
Some notes

- $sr_{target\_highest\_slotid} \leq sr_{highest\_slotid}$
- Since dynamic slot allocation is not a mandatory feature (but a really useful one), then servers SHOULD ensure that for clients that don’t support dynamic slot allocation, $sr_{highest\_slotid} \geq csr_{fore\_chan\_attrs.ca\_maxrequests}-1$ (see CREATE_SESSION).
Sounds easy. Where’s the catch?

- Asynchronous nature of communication means that the client and server need to be careful when updating the values for `sr_highest_slotid`, `sr_target_highest_slotid`.
  - SEQUENCE requests/replies on different slots can be reordered w.r.t. each other.
How does reordering create problems?

- Client sees incorrect limits:

  ![Diagram showing reordering issues]

  Server
  
  Client
  
  T=10  H=15
  T=5   H=10
  T=5   H=5
  H=5
  H=5
  H=10
How does reordering create problems?

- Server sees incorrect client load:

  ![Diagram showing server and client load](image-url)
When can `sr_highest_slotid` decrease?

- After changing `sr_target_highest_slotid`.
  - Need to know that the client is not trying to replay any requests on those slots
  - Check `sa_highest_slotid`.
    - But what if it was reordered?
How does reordering create problems?

- Server retires `sr_highest_slotid` too early:

```

Client

H=6  H=7  H=7 (replay)
```

```

Server

H=7
T=7  H=7

T=6  H=6
T=6  H=6
```

When can `sr_highest_slotid` decrease?

- After changing `sr_target_highest_slotid`.
  - Need to know that the client is not trying to replay any requests on those slots
  - Check `sa_highest_slotid`.
    - But what if it was reordered?

- Solve reordering problem by checking `sa_highest_slotid` only on slots on which the new `sr_target_highest_slotid` have been sent.
  - Server needs to track value of `sr_target_highest_slotid` for each slot.
When can sr\_highest\_slotid decrease

- Alternative server strategy is to only grow the window using sr\_target\_highest\_slotid mechanism.
  - Use CB\_RECALL\_SLOT to tell the client to shrink the window
  - Problem is that only solves the reordering issues for server highest slotid limits.
Protocol nits…

- RFC5661 does not say what happens to the sequence id for a “new” slot, when the server raises `sr_highest_slotid`.
  - Should it be initialised to ‘0’ on the server?
    - Reordering corner cases: client may fail to see slot being retired and then reinstated...
    - Alternative is to allow any initial value.
  - Need an errata…
Implementation: client

- Linux 3.7 upstream NFSv4.1 client and newer implements dynamic slot allocation on the forward channel.
  - Supports CB_RECALL_SLOT
  - Client will generate extra SEQUENCE ops in order to satisfy lower target highest slotid.
  - Implements simple smoothing to avoid re-ordering issues w.r.t. highest slotid and target.
Implementation: server

  - Implements basic client-driven policy
    - grow the number of slots by $\frac{1}{4}$ when $sa_{\text{highest_slotid}} \geq sr_{\text{target_highest_slotid}}$
    - Shrink slot table when $sa_{\text{highest_slotid}}$ is decreasing
  - Global maximum number of slots.
  - Smoothing used to avoid $sa_{\text{highest_slotid}}$ reordering issues.
Thank you