NetApp®

Go further, faster®

LAYOUTRETURN: error codes

Tom Haynes

LAYOUTRETURN communication of errors

- Object layouts uses lrf_body
  - Opaque array
- Blocks layouts states lrf_body MUST be NULL
- File layout states nada

How to handle errors

- Each layout type uses lrf_body
  - If NULL, no error
  - Else it describes errors encountered between DS and clients
- MDS decides what to do, if anything
  - Might be an expected state, i.e., fencing
- MDS might decide to avoid using that DS in layouts
  - If DS is down, why hand it out

NetApp

Error returns for File layouts

- Set an error code per DS
  - NFS4_OK
  - NFS4ERR_NXIO means the client had a non-transient connectivity issue with the DS
  - Anything else indicates an I/O error experienced by the client (could have been any OP)
- Provide the specific operation which had the error

Client’s responsibilities

- Has to realize server may not have any choice
  - Might be a limited set of DSes
- Has to realize what it considers a fatal error, might not be an error
  - MDS has fenced it off
- If new layout has a DS with existing issues
  - SHOULD periodically try pNFS access to see if the issue has healed itself
  - Can return the layout and ask for READ/WRITE access via the MDS

Server's responsibilities

- For existing files
  - SHOULD not return layouts with the problematic DSes to that client for some time
    - I.e., force the client into READ/WRITE I/O through the MDS
    - See if problem heals after time
  - Track whether other clients report an issue
  - Could restripe the files being requested
- For newly created files
  - SHOULD not use the problematic DS with that client if at all possible for some time
    - See if the problem heals after time

New XDR

```
 const LAYOUT4_RET_REC_FILE          = 1;
const LAYOUT4_RET_REC_FSID          = 2;
const LAYOUT4_RET_REC_ALL           = 3;

enum layoutreturn_type4 {
     LAYOUTRETURN4_FILE = LAYOUT4_RET_REC_FILE,
     LAYOUTRETURN4_FSID = LAYOUT4_RET_REC_FSID,
     LAYOUTRETURN4_ALL  = LAYOUT4_RET_REC_ALL,
};

struct layoutreturn_file4 {
      offset4         lrf_offset;
      length4         lrf_length;
      stateid4        lrf_stateid;
      /* layouttype4 specific data */
      opaque          lrf_body<>;
};

struct layoutreturn_device_error4 {
      deviceid4     lrde_deviceid;
      nfsstat4      lrde_status;
      nfs_opnum4    lrde_opnum;
};

struct layoutreturn_file_error4 {
      layoutreturn_file4            lrfe_layout;
      layoutreturn_device_error4    lrfe_device<>;
};

union layoutreturn4 switch(layoutreturn_type4 lr_returntype) {
      case LAYOUTRETURN4_FILE:
              layoutreturn_file4            lr_layout;
      default:
              void;
};
```