pNFS Access Permissions Check

IETF 76 NFSv4 WG Meeting
November 11, 2009

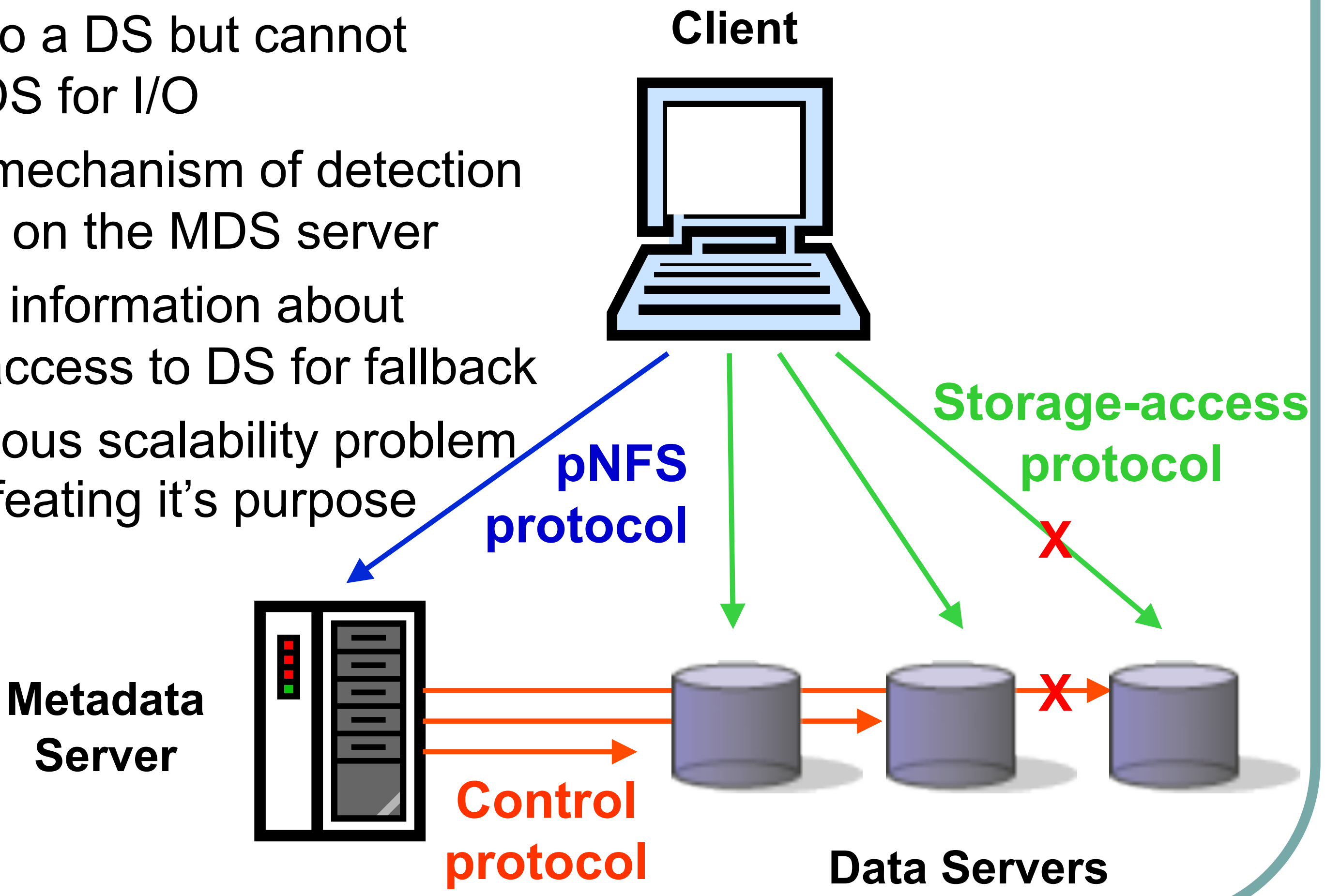
Sorin Faibish sfaibish@emc.com
David Black
Mike Eisler

Outline

- Problem Statement
- Protocol Gaps
- Possible Remedies
- Error Reporting
- Permission Checks
- Implementation Proposals
- Questions and Discussion

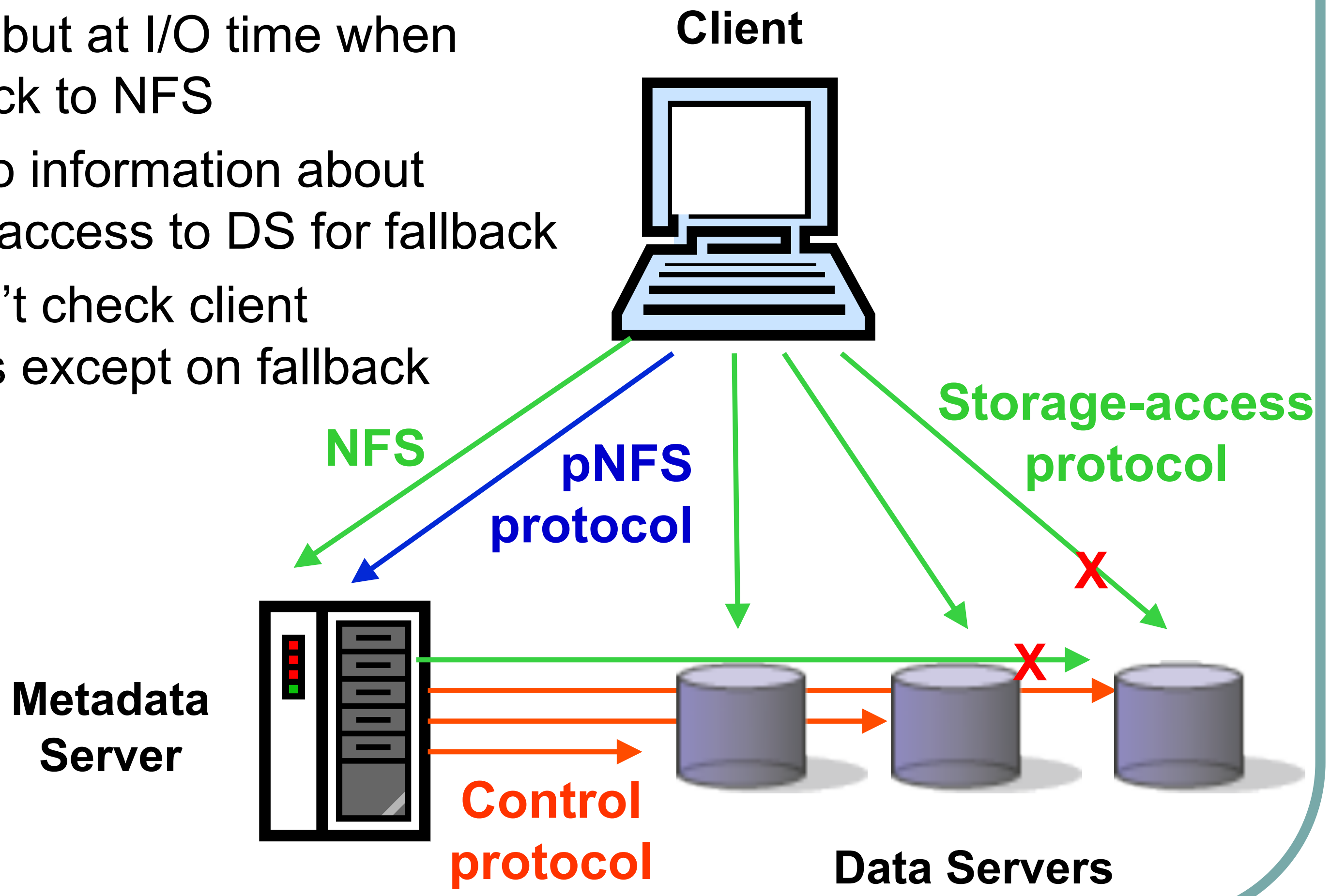
Problem Statement

- pNFS clients can receive from MDS valid layout to a DS but cannot access the DS for I/O
- There is no mechanism of detection or correction on the MDS server
- MDS has no information about permission access to DS for fallback
- This is a serious scalability problem for pNFS defeating it's purpose



Problem Statement (cont.)

- Permission denial is not detected at mount time but at I/O time when client fallback to NFS
- MDS has no information about permission access to DS for fallback
- MDS doesn't check client permissions except on fallback detection



Protocol gaps

1. There is no error report mechanism for client and MDS for permission access issues
2. MDS can deliver valid layout to clients that have no permission to a DS without check
3. There is no correction mechanism of the MDS to recall a layout and remove the DS with issues
4. The permission problem is not reported at mount time (/ is pNFS mounted) and may have a performance penalty during I/O
5. No guarantees that fallback to MDS will succeed
6. pNFS specification does not address the protocol between the MDS and DS

Proposed Remedies (protocol)

1. A protocol change is needed as both server and client needs modifications. Optimizations don’t work: ***draft-faibish-nfsv4-pnfs-access-permissions-check-01***
2. Add access permission error reporting to both client and server using new LAYOUTRETURN command
3. Add a new LAYOUTRECALL CB command and LAYOUTRETURN command requiring the client to perform a permission check and return all layouts for DS with permission issues
4. Leave the detection of permission problem condition as a recommendation for the server implementation.
5. On detection the server will remove the DS from the valid DS list configuration or flag it as inaccessible and will recall all the layouts that include that DS and send new layouts excluding the DS to clients

Error Reporting (draft)

1. Add client error reporting to LAYOUTRETURN opaque for permission access denial before fallback to NFS – enhance 4.1 same as object layout (draft 3.1).
2. Introduce a new LAYOUTRETURN_DEVICE command for which the client returns all the layouts for the denied device and report a new error case (draft 3.2).
3. Same error reporting mechanism will be used in combination with the new CB_LAYOUTRECALL/ LAYOUTRETURN commands after agreement of the WG

Permission Checks (proposed)

1. Detection of the problem will be left as an implementation (count of fallbacks to NFS for a certain DS/device)
2. The protocol will define the permission checks and the response to permission issues. A new LAYOUT command will be introduced in 4.2 as well as a new error report mechanism
3. A new CB_LAYOUTRECALL command will be introduced in 4.2 asking for permission check different from connectivity issues
4. A new LAYOUTRETURN command will be introduced in 4.2 returning all the layouts for a given DS/device before a fallback to NFS
5. On fallback to NFS the server will send a new layout to client excluding the DS/device with permission denial.
6. Special cases of compression/encryption will be left outside the scope of permission checks and related fallback to NFS will be differentiated from permission checks

Permission Checks (alternatives)

- Server unilaterally send a CB_LAYOUTRECALL of all the layouts on the device with the permission issue to all clients that have valid layouts on that DS and send a new layout on next LAYOUTGET command of the client
- Server perform a permission check of himself to access to the DS and log permission error and remove DS/device from configuration
- Server sends a new CB_LAYOUTRECALL asking for client permission check to a specific DS/device and ask return of the layout on that device on error
- A client can send a LAYOUTRETURN command for the layout on DS to which it has a permission issue and fallback to NFS for that device
- Server will differentiate from a normal fallback or a permission related fallback of many clients.

Implementation: error report

- For file layout type define the opaque body:

```
struct nfsv4_1_file_layoutreturn4 {
        deviceid4      lrf_deviceid;
        nfsstat4       lrf_status;
    };.
```

  MDS will check size of the opaque lrf_body if non-zero=error

- For the block layout type the specific strucutre:

```
struct pnfs_block_layoutreturn4 {
       deviceid4      lrf_deviceid;
       nfsstat4       lrf_status;
};
```

  MDS will check size of the opaque lrf_body if non-zero=error

- For the object layout type opaque already exist see:

  draft-ietf-nfsv4-pnfs-obj-12

Can be added as an extension of object layout to file and block in 4.1

Implementation: new LAYOUTRETURN

- Add new constant:

    const LAYOUT4_RET_REC_DEVICE = 4;

- Add a new error code:

    NFS4ERR_DEVICE_PERM_DENY

- Add new LAYOUTRETURN layoutreturn_type4:

    LAYOUT4_RET_REC_DEVICE_NO_ACCESS

- To address the backward compatibility may require a client to do two layout return operations to deal with servers that don't understand the new layoutreturn_type4

We propose this implementation for 4.2

Questions & Discussions

- Is the permission check needed or is error reporting enough?
- Is this issue a protocol change or just an optimization? How real is this usecase?
- Are the proposed protocol changes too complex for the pNFS protocol? Should we find an easier solution?
- Are new layout commands needed or should modify existing commands?
- Will you support this draft and review it?

Acknoledgements

Thanks to:

- Jason Glasgow defined the implementation details and reviewed the draft
- Mike Eisler for brainstorming the alternative implementation decisions
- David Black for help with my first draft publication and extremely thorough review