pNFS with IPv6

The discussion was related to some findings during the interop tests of pNFS file layout with IPv6.

The MDS and DS’s were configured with both IPv6 and IPv4 IP addresses as well as the clients. The mount and MD operations were performed over the IPv6 but we noticed that the I/O operations were done over IPv4. When we configured the DS to support only IPv6 the clients hang.

Chuck notes that a couple of things need to be considered:

a) On non-pNFS mounts, clients are able to force a particular transport and protocol family choice via the "proto=" mount option. I think some in the community might prefer to retain that ability for pNFS, but I haven't considered the implications carefully.

Trond opinion is that when talking to a single server, this makes partial sense (it is mainly about making the transition from non-auto-negotiating NFSv2/v3 to auto-negotiating NFSv4 easy). When talking to an NFSv4.1 metadata server plus a myriad pNFS data servers, it makes less sense. There is no commitment to provide that kind of functionality to Linux administrators at this point. Not until someone can show an usecase where it makes sense to override the automatic probing.

b) Clients should not be _required_ to use the same transport and protocol family for the DS and MDS. To wit: for pNFS file mode, RDMA will probably be appropriate for DS activity, but may not be supported for MDS operations.

Trond argued that the protocol allows the server to advertise an array of struct netaddr4 for each DS. The struct netaddr4 contains both a netid and a universal address. The client may then negotiate RDMA upon setting up the session, if the data server supports it.

Currently, the MDS servers can return only one IPv4 and one IPv6 address per DS. Bruce suggested using multipath to return a richer list that could contain more information. Others in the room thought that was a good suggestion.

But this is not a protocol limitation. It is a limitation of the current generation of servers implementation.

If the server returns a list, perhaps the list should be priority ordered from the server's point of view, and perhaps there should be an RFC-proscribed algorithm for the client to choose which address and transport to use for its DS operations.

However, client implementations should be free to ignore the server's ordering. The most reasonable thing to do is to have them auto-probe based on the server's ordering. Anything else is likely to require us to set up an extra user interface in order to determine policy. Currently there is no commitment of the clients to do this until we hear valid usecases.

Sorin’s usecase: have the servers return a list containing addresses in both protocol families; disable one set of families on the DSes. In this case, it was the family that was at the top of the server's list. The current client implementations do not attempt to discover a working connection to the DSes by walking that list.

Trond explained that in the current implementation if the DS supports only IPv6 the client will hang. In the current implementation the Linux client will access the DS only using IPv4. The reason is that the IP address is also included in the layout received by the client from the MDS. If the DS IP sent to the client is IPv4 one then the client is able to perform the I/Os to the DS. Today the server implementation sends only one IP in the layout for each DS from the IP addresses list. If the MDS selects the IPv4 first the client will perform I/Os to the DS. But if the MDS sends IPv6 in the layout the client hangs and it doesn’t retry to the send the I/O to the MDS. This is simply missing functionality in the current Linux client implementation not a protocol issue.

In the previous cthon we tested IPv6 with pNFS block server only and it worked well as the I/O’s are done to SCSI devices via iSCSI target that supports any IP that the client uses and as such all the MD operations were done to the MDS using the IPv6 the client mounted IPv6. This is not same case with the file layout that includes the IP in the layout explicitly.

As a result of these findings we discussed the correct behavior according to the RFC will be three fold:

- if the client cannot access the DS using the IPv6 address it should fall back to use IPv6 on MDS
- if the DS supports both IPv4 and IPv6 and the client can mount the DS using IPv6 it should be able to use IPv6 for I/O to DS’s
- the client will mount using IPv6 the MDS but there is no guarantee about what IP will use to access the DS

Currently the server implementations are such that they send only one IP in the layout using the list of announced IP’s of the DS’s. As a result of the fact that the client only uses a certain IP address creates a problem if the layout has an address not supported by the client. There is no way to find out if the client can access a DS using a certain type of IP to the MDS. The result (in Linux) is that if the layout contains a IPv4 address it works and if not it doesn’t work.

Chuck and Bruce proposed to recommend the servers to send a list of IP addresses that the DS’s support and the client will select the one it supports. An alternative solution is for the client to do a layoutreturn with connectivity error.

Trond does not think that any of this is a real problem. These are limitations that we deliberately imposed on ourselves in order to be able to ship pNFS faster. We will fix them up when we're done with integrating the basic code into our respective kernels.

In conclusion we could modify/enhance the 4.1 protocol to include implementation recommendations to either including a list of IPs of each DS or use a layoutreturn and a new error case similar to the Permission Access draft to communicate to the server the preferred IP for the client and the server to resend the layout using the preferred IP address for the client if there are multiple IP types. The problem can become more complicated if the client is using session trunking and/or clientid trunking when DS’s support multiple IP types.