NFSv4
Internet-Draft
Intended status: Informational
Expires: September 16, 2014

D. Noveck, Ed.

P. Shivam
C. Lever
B. Baker
ORACLE

March 15, 2014

NFSv4 migration: Implementation experience and spec issues to resolve draft-ietf-nfsv4-migration-issues-05

Abstract

The migration feature of NFSv4 provides for moving responsibility for a single filesystem from one server to another, without disruption to clients. Recent implementation experience has shown problems in the existing specification for this feature. This document discusses the issues which have arisen, explores the options available for curing the issues, and explains the choices made in updating the NFSv4.0 and NFSv4.1 specifications, to address migration.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 16, 2014.

Copyright Notice

Table of Contents

1. Introduction

This document is in the informational category, and while the facts it reports may have normative implications, any such normative significance reflects the readers' preferences. For example, we may report that the reboot of a client with migrated state results in state not being promptly cleared and that this will prevent granting of conflicting lock requests at least for the lease time, which is a fact. While it is to be expected that client and server implementers will judge this to be a situation that is best avoided, the judgment as to how pressing this issue should be considered is a judgment for the reader, and eventually the nfsv4 working group to make.

We do explore possible ways in which such issues can be avoided, with minimal negative effects, given that the working group has decided to address these issues, but the choice of exactly how to address these is best given effect in one or more standards-track documents and/or errata.

This document focuses on NFSv4.0, since that is where the majority of implementation experience has been. Nevertheless, there is discussion of the implications of the NFSv4.0 experience for migration in NFSv4.1, as well as discussion of other issues with regard to the treatment of migration in NFSv4.1.

2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

In the context of this informational document, these normative keywords will always occur in the context of a quotation, most often direct but sometimes indirect. The context will make it clear whether the quotation is from:

- The current definitive definition of the NFSv4.0 protocol, whether that is the original NFSv4.0 specification [RFC3530], or its expected successor [RFC3530bis].

As the identity of that document may change during the lifetime of this document, we will often refer to the current or pending definition of NFSv4.0 and quote from portions of the documents that are identical among all existing drafts. Given that RFC3530 and all RFC3530bis drafts agree as to the issues under discussion, this should not cause undue difficulty. Note that to simplify document maintenance, section names rather than section numbers are used when referring to sections in existing documents so that only minimal changes will be necessary as the identity of the document defining NFSv4.0 changes.

- The current definitive definition of the NFSv4.1 protocol [RFC5661].

- A proposed or possible text to serve as a replacement for the current definitive document text. Sometimes, a number of possible alternative texts may be listed and benefits and detriments of each examined in turn.

3. NFSv4.0 Implementation Experience

3.1. Implementation issues

Note that the examples below reflect current experience which arises from clients implementing the recommendation to use different nfs_client_id4 id strings for different server addresses, i.e. using what is later referred to herein as the "non-uniform client-string approach."

This is simply because that is the experience implementers have had. The reader should not assume that in all cases, this practice is the source of the difficulty. It may be so in some cases but clearly it is not in all cases.

3.1.1. Failure to free migrated state on client reboot

The following sort of situation has proved troublesome:

- A client C establishes a clientid4 C1 with server ABC specifying an nfs_client_id4 with id string value "C-ABC" and boot verifier 0x111.

- The client begins to access files in filesystem F on server ABC, resulting in generating stateids S1, S2, etc. under the lease for clientid C1. It may also access files on other filesystems on the same server.

- The filesystem is migrated from server ABC to server XYZ. When transparent state migration is in effect, stateids S1 and S2 and clientid4 C1 are now available for use by client C at server XYZ.

- Client C reboots and attempts to access data on server XYZ, whether in filesystem F or another. It does a SETCLIENTID with an nfs_client_id4 with id string value "C-XYZ" and boot verifier 0x112. There is thus no occasion to free stateids S1 and S2 since they are associated with a different client name and so lease expiration is the only way that they can be gotten rid of.

Note here that while it seems clear to us in this example that C-XYZ and C-ABC are from the same client, the server has no way to determine the structure of the "opaque" id string. In the protocol, it really is treated as opaque. Only the client knows which nfs_client_id4 values designate the same client on a different server.

3.1.2. Server reboots resulting in a confused lease situation

Further problems arise from scenarios like the following.

- Client C talks to server ABC using an nfs_client_id4 id string such as "C-ABC" and a boot verifier v1. As a result, a lease with clientid4 c.i is established: {v1, "C-ABC", c.i}.

- fs_a1 migrates from server ABC to server XYZ along with its state. Now server XYZ also has a lease: {v1, "C-ABC", c.i}.

- Server ABC reboots.

- Client C talks to server ABC using an nfs_client_id4 id string such as "C-ABC" and a boot verifier v1. As a result, a lease with clientid4 c.j is established: {v1, "C-ABC", c.j}.

- fs_a2 migrates from server ABC to server XYZ. Now server XYZ also has a lease: {v1, "C-ABC", c.j}.

- Now server XYZ has two leases that match {v1, "C-ABC", *}, when the protocol clearly assumes there can be only one.

Note that if the client used "C" (rather than "C-ABC") as the nfs_client_id4 id string, the exact same situation would arise.

One of the first cases in which this sort of situation has resulted in difficulties is in connection with doing a SETCLIENTID for callback update.

The SETCLIENTID for callback update only includes the nfs_client_id4, assuming there can only be one such with a given nfs_client_id4 value. If there were multiple, confirmed client records with identical nfs_client_id4 id string values, there would be no way to map the callback update request to the correct client record. Apart from the migration handling specified in [RFC3530] and [RFC3530bis], such a situation cannot arise.

One possible accommodation for this particular issue that has been used is to add a RENEW operation along with SETCLIENTID (on a callback update) to disambiguate the client.

When the client updates the callback info to the destination, the client would, by convention, send a compound like this:

{ RENEW clientid4, SETCLIENTID nfs_client_id4,verf,cb }

The presence of the clientid4 in the compound would allow the server to differentiate among the various leases that it knows of, all with the same nfs_client_id4 value.

While this would be a reasonable patch for an isolated protocol weakness, interoperable clients and servers would require that the protocol truly be updated to allow such a situation, specifically that of multiple clientid4's with the same nfs_client_id4 value. The protocol is currently designed and implemented assuming this cannot happen. We need to either prevent the situation from happening, or fully adapt to the possibilities which can arise. See Section 4 for a discussion of such issues.

3.1.3. Client complexity issues

Consider the following situation:

- There are a set of clients C1 through Cn accessing servers S1 through Sm. Each server manages some significant number of filesystems with the filesystem count L being significantly greater than m.

- Each client Cx will access a subset of the servers and so will have up to m clientids, which we will call Cxy for server Sy.

- Now assume that for load-balancing or other operational reasons, numbers of filesystems are migrated among the servers. As a result, each client-server pair will have up to m clientids and each client will have up to m**2 clientids. If we add the possibility of server reboot, the only bound on a client's clientid count is L.

Now, instead of a clientid4 identifying a client-server pair, we have many more entities for the client to deal with. In addition, it isn't clear how new state is to be incorporated in this structure.

The limitations of the migrated state (inability to be freed on reboot) would argue against adding more such state but trying to avoid that would run into its own difficulties. For example, a single lockowner string presented under two different clientids would appear as two different entities.

Thus we have to choose between:

- indefinite prolongation of foreign clientids even after all transferred state is gone.
- having multiple requests for the same lockowner-string-named entity carried on in parallel by separate identically named lockowners under different clientid4's
- Adding serialization at the lock-owner string level, in addition to that at the lockowner level.

In any case, we have gone (in adding migration as it was described) from a situation in which

- Each client has a single clientid4/lease for each server it talks to.
- Each client has a single nfs_client_id4 for each server it talks to.
- Every state id can be mapped to an associated lease based on the server it was obtained from.

To one in which

- Each client may have multiple clientid4's for a single server.
- For each stateid, the client must separately record the clientid4 that it is assigned to, or it must manage separate "state blobs" for each fsid and map those to clientid4's.
- Before doing an operation that can result in a stateid, the client must either find a "state blob" based on fsid or create a new one, possibly with a new clientid4.
- There may be multiple clientid4's all connected to the same server and using the same nfs_clientid4.

This sort of additional client complexity is troublesome and needs to be eliminated.

3.2. Sources of Protocol difficulties

3.2.1. Issues with nfs_client_id4 generation and use

The current definitive definitions of the NFSv4.0 protocol, [RFC3530] and [RFC3530bis] both agree. The section entitled "Client ID" says:

> The second field, id is a variable length string that uniquely defines the client.

There are two possible interpretations of the phrase "uniquely defines" in the above:

- The relation between strings and clients is a function from such strings to clients so that each string designates a single client.

- The relation between strings and clients is a bijection between such strings and clients so that each string designates a single client and each client is named by a single string.

The first interpretation would make these client-strings like phone numbers (a single person can have several) while the second would make them like social security numbers.

Endless debate about the true meaning of "uniquely defines" in this context is quite possible but not very helpful. The following points should be noted though:

- The second interpretation is more consistent with the way "uniquely defines" is used elsewhere in the spec.

- The spec as now written intends the first interpretation (or is internally inconsistent). In fact, it recommends, although it doesn't "RECOMMEND" that a single client have at least as many client-strings as server addresses that it interacts with. It says, in the third bullet point regarding construction of the string (which we shall henceforth refer to as client-string-BP3):

  > The string should be different for each server network address that the client accesses, rather than common to all server network addresses.

- If internode interactions are limited to those between a client and its servers, there is no occasion for servers to be concerned with the question of whether two client-strings designate the same

client, so that there is no occasion for the difference in interpretation to matter.

- When transparent migration of client state occurs between two servers, it becomes important to determine when state on two different servers is for the same client or not, and this distinction becomes very important.

Given the need for the server to be aware of client identity with regard to migrated state, either client-string construction rules will have to change or there will be a need to get around current issues, or perhaps a combination of these two will be required. Later sections will examine the options and propose a solution.

One consideration that may indicate that this cannot remain exactly as it is today has to do with the fact that the current explanation for this behavior is not correct. The current definitive definitions of the NFSv4.0 protocol, [RFC3530] and [RFC3530bis] both agree. The section entitled "Client ID" says:

> The reason is that it may not be possible for the client to tell if the same server is listening on multiple network addresses. If the client issues SETCLIENTID with the same id string to each network address of such a server, the server will think it is the same client, and each successive SETCLIENTID will cause the server to begin the process of removing the client's previous leased state.

In point of fact, a "SETCLIENTID with the same id string" sent to multiple network addresses will be treated as all from the same client but will not "cause the server to begin the process of removing the client's previous leased state" unless the server believes it is a different instance of the same client, i.e. if the id string is the same and there is a different boot verifier. If the client does not reboot, the verifier should not change. If it does reboot, the verifier will change, and the server should "begin the process of removing the client's previous leased state.

The situation of multiple SETCLIENTID requests received by a server on multiple network addresses is exactly the same, from the protocol design point of view, as when multiple (i.e. duplicate) SETCLIENTID requests are received by the server on a single network address. The same protocol mechanisms that prevent erroneous state deletion in the latter case prevent it in the former case. There is no reason for special handling of the multiple-network-appearance case, in this regard.

3.2.2. Issues with lease proliferation

It is often felt that this is a consequence of the client-string construction issues, and it is certainly the case that the two are closely connected in that non-uniform client-strings make it impossible for the server to appropriately combine leases from the same client.

However, even where the server could combine leases from the same client, it needs to be clear how and when it will do so, so that the client will be prepared. These issues will have to be addressed at various places in the spec.

This could be enough only if we are prepared to do away with the "should" recommending non-uniform client-strings and replace it with a "should not" or even a "SHOULD NOT". Current client implementation patterns make this an unpalatable choice for use as a general solution, but it is reasonable to "RECOMMEND" this choice for a well-defined subset of clients. One alternative would be to create a way for the server to infer from client behavior which leases are held by the same client and use this information to do appropriate lease mergers. Prototyping and detailed specification work has shown that this could be done but the resulting complexity is such that a better choice is to "RECOMMEND" use of the uniform client-string approach for clients supporting the migration feature.

Because of the discussion of client-string construction in [RFC3530] and [RFC3530bis], most existing clients implement the non-uniform client-string approach. As a result, existing servers may not have been tested with clients implementing uniform client-strings. As a consequence, care must be taken to preserve interoperability between UCS-capable clients and servers that don't tolerate uniform client strings for one reason or another.

4. Issues to be resolved in NFSv4.0

4.1. Possible changes to nfs_client_id4 client-string

The fact that the reason given in client-string-BP3 is not valid makes the existing "should" insupportable. We can't either

- Keep a reason we know is invalid.
- Keep saying "should" without giving a reason.

What are often presented as reasons that motivate use of the non-uniform approach always turn out to be cases in which, if the uniform approach were used, the server will treat a client which accesses

that server via two different IP addresses as part of a single client, as it in fact is. This may be disconcerting to a client unaware that the two IP addresses connect to the same server. This is not a reason to use the non-uniform approach but is better thought of as an illustration of the fact that those using the uniform approach need to be aware of the possibility of server trunking and its effect on server behavior.

If it is possible to reliably infer the existence of trunking of server IP addresses from observed server behavior, use of the uniform approach would be more desirable, although compatibility issues would have to be dealt with.

An alternative to having the client infer the existence of trunking of IP server addresses, is to make this information available to the client directly. See Section 4.3 for details.

It is always possible that a valid new reason will be found, but so far none has been proposed. Given the history, the burden of proof should be on those asserting the validity of a proposed new reason.

So we will assume for now that the "should" will have to go. The question is what to replace it with.

- We can't say "MUST NOT", despite the problems this raises for migration since this is pretty late in the day for such a change. Many currently operating clients obey the existing "should". Similar considerations would apply for "SHOULD NOT" or "should not".

- Dropping client-string-BP3 entirely is a possibility but, given the context and history, it would just be a confusing version of "SHOULD NOT".

- Using "MAY" would clearly specify that both ways of doing this are valid choices for clients and that servers will have to deal with clients that make either choice.

- This might be modified by a "SHOULD" (or even a "MUST") for particular groups of clients.

- There will have to be some text explaining why a client might make either choice but, except for the particular cases referred to above, we will have to make sure that it is truly descriptive, and not slanted in either direction.

4.2. Possible changes to handle differing nfs_client_id4 string values

Given the difficulties caused by having different nfs_client_id4 client-string values for the same client, we have two choices:

- Deprecate the existing treatment and basically say the client is on its own doing migration, if it follows it.
- Introduce a way of having the client provide client identity information to the server, if it can be done compatibly while staying within the bounds of v4.0.

4.3. Possible changes to add a new operation

It might be possible to return server-identity information to the client, just as is done in NFSv4.1 by the response to the EXCHANGE_ID operation. This could be done by a SETCLIENTID_PLUS optional operation, which acts like SETCLIENTID, except that it returns server identity information. Such information could be used by clients, making it possible to for them to be aware of server trunking relationships, rather than having to infer them from server behavior.

It has been generally thought that protocol extensions such as this are not appropriate in bis documents and other documents updating NFSv4 protocol definition RFC's. However, it is argued in [NFS-ext] that protocol extensions, similar to those allowed between minor versions, should be acceptable to correct mistakes within a minor version.

A decision to adopt this approach will require considerable nfsv4 working group discussion and would probably best be effected by means of a standards-track document laying out a modified NFSv4 extension/ versioning model applying to all minor versions, as has been proposed.

In view of the time to effect such changes, this approach is not likely to be adopted in an RFC updating [RFC3530] or [RFC3530bis], such as [migr-v4.0-update]. Still, it is worth keeping in mind, if implementers have difficulties inferring trunking relationships using the techniques discussed there.

4.4. Other issues within migration-state sections

There are a number of issues where the existing text is unclear and/ or wrong and needs to be fixed in some way.

- Lack of clarity in the discussion of moving clientids (as well as stateids) as part of moving state for migration.

- The discussion of synchronized leases is wrong in that there is no way to determine (in the current spec) when leases are for the same client and also wrong in suggesting a benefit from leases synchronized at the point of transfer. What is needed is merger of leases, which is necessary to keep client complexity requirements from getting out of hand.

- Lack of clarity in the discussion of LEASE_MOVED handling, including failure to fully address situations in which transparent state migration did not occur.

4.5. Issues within other sections

There are a number of cases in which certain sections, not specifically related to migration, require additional clarification. This is generally because text that is clear in a context in which leases and clientids are created in one place and live there forever may need further refinement in the more dynamic environment that arises as part of migration.

Some examples:

- Some people are under the impression that updating callback endpoint information for an existing client, as used during migration, may cause the destination server to free existing state. There need to be additions to clarify the situation.

- The handling of the sets of clientid4's maintained by each server needs to be clarified. In particular, the issue of how the client adapts to the presumably independent and uncoordinated clientid4 sets needs to be clearly addressed

- Statements regarding handling of invalid clientid4's need to be clarified and/or refined in light of the possibilities that arise due to lease motion and merger.

- Confusion and lack of clarity about NFS4ERR_CLID_INUSE.

5. Proposed resolution of NFSv4.0 protocol difficulties

This section lists the changes which we believe are necessary to resolve the difficulties mentioned above. Such change, along with other clarifications found to be desirable during drafting and review are contained in [migr-v4.0-update].

5.1. Proposed changes: nfs_client_id4 client-string

We propose replacing client-string-BP3 with the following text and adding the following proposed to provide implementation guidance.

> The string MAY be different for each server network address that the client accesses, rather than common to all server network addresses.

In addition, given the importance of the issue of client identity and the fact that both client string-approaches are to be considered valid, a greatly expanded treatment of client identity desirable. It should have the following major elements.

- It should fully describe the consequences of making the string different for each network address (the non-uniform client-string approach) and of making it the same for all network addresses (the uniform client string approach).
- It should give helpful guidance about the factors that might affect client implementation choice between these approaches.
- It should describe the compatibility issues that might cause servers to be incompatible with the uniform approach and give guidance about dealing with these.
- It should describe how a client using the uniform approach might use server behavior to determine server address trunking patterns.
- It should present a clearer and more complete set of recommendations to guide client string construction.

5.2. Proposed changes: merged (vs. synchronized) leases

The current definitive definitions of the NFSv4.0 protocol, [RFC3530] and [RFC3530bis] both agree. The section entitled "Migration and State" says:

> As part of the transfer of information between servers, leases would be transferred as well. The leases being transferred to the new server will typically have a different expiration time from those for the same client, previously on the old server. To maintain the property that all leases on a given server for a given client expire at the same time, the server should advance the expiration time to the later of the leases being transferred or the leases already present. This allows the client to maintain lease renewal of both classes without special effort:

There are a number of problems with this and any resolution of our difficulties must address them somehow.

- The current v4.0 spec recommends that the client make it essentially impossible to determine when two leases are from "the same client".

- It is not appropriate to speak of "maintain[ing] the property that all leases on a given server for a given client expire at the same time", since this is not a property that holds even in the absence of migration. A server listening on multiple network addresses may have the same client appear as multiple clients with no way to recognize the client as the same.

- Even if the client identity issue could be resolved, advancing the lease time at the point of migration would not maintain the desired synchronization property. The leases would be synchronized until one of them was renewed, after which they would be unsynchronized again.

To avoid client complexity, we need to have no more than one lease between a single client and a single server. This requires merger of leases since there is no real help from synchronizing them at a single instant.

For the uniform approach, the destination server would simply merge leases as part of state transfer, since two leases with the same nfs_client_id4 values must be for the same client.

We have made the following decisions as far as proposed normative statements regarding for state merger. They reflect the facts that we want to support fully migration support in the simplest way possible and that we can't say MUST since we have older clients and servers to deal with.

- Clients SHOULD use the uniform client-string approach in order to get good migration support.

- Servers SHOULD provide automatic lease merger during state migration so that clients using the uniform id approach get the support automatically.

If the clients and the servers obey the SHOULD's, having more than a single lease for a given client-server pair will be a transient situation, cleaned up as part of adapting to use of migrated state.

Since clients and servers will be a mixture of old and new and because nothing is a MUST we have to ensure that no combination will

show worse behavior than is exhibited by current (i.e. old) clients and servers.

5.3. Other proposed changes to migration-state sections

5.3.1. Proposed changes: Client ID migration

The current definitive definitions of the NFSv4.0 protocol, [RFC3530] and [RFC3530bis] both agree. The section entitled "Migration and State" says:

> In the case of migration, the servers involved in the migration of a filesystem SHOULD transfer all server state from the original to the new server. This must be done in a way that is transparent to the client. This state transfer will ease the client's transition when a filesystem migration occurs. If the servers are successful in transferring all state, the client will continue to use stateids assigned by the original server. Therefore the new server must recognize these stateids as valid. This holds true for the client ID as well. Since responsibility for an entire filesystem is transferred with a migration event, there is no possibility that conflicts will arise on the new server as a result of the transfer of locks.

This poses some difficulties, mostly because the part about "client ID" is not clear:

- It isn't clear what part of the paragraph the "this" in the statement "this holds true ..." is meant to signify.

- The phrase "the client ID" is ambiguous, possibly indicating the clientid4 and possibly indicating the nfs_client_id4.

- If the text means to suggest that the same clientid4 must be used, the logic is not clear since the issue is not the same as for stateids of which there might be many. Adapting to the change of a single clientid, as might happen as a part of lease migration, is relatively easy for the client.

We have decided that it is best to address this issue as follows:

- Make it clear that both clientid4 and nfs_client_id4 (including both id string and boot verifier) are to be transferred.

- Indicate that the initial transfer will result in the same clientid4 after transfer but this is not guaranteed since there may conflict with an existing clientid4 on the destination server and because lease merger can result in a change of the clientid4.

5.3.2. Proposed changes: Callback re-establishment

The current definitive definitions of the NFSv4.0 protocol, [RFC3530] and [RFC3530bis] both agree. The section entitled "Migration and State" says:

> A client SHOULD re-establish new callback information with the new server as soon as possible, according to sequences described in sections "Operation 35: SETCLIENTID - Negotiate Client ID" and "Operation 36: SETCLIENTID_CONFIRM - Confirm Client ID". This ensures that server operations are not blocked by the inability to recall delegations.

The above will need to be fixed to reflect the possibility of merging of leases,

5.3.3. Proposed changes: NFS4ERR_LEASE_MOVED rework

The current definitive definitions of the NFSv4.0 protocol, [RFC3530] and [RFC3530bis] both agree. The section entitled "Notification of Migrated Lease" says:

> Upon receiving the NFS4ERR_LEASE_MOVED error, a client that supports filesystem migration MUST probe all filesystems from that server on which it holds open state. Once the client has successfully probed all those filesystems which are migrated, the server MUST resume normal handling of stateful requests from that client.

There is a lack of clarity that is prompted by ambiguity about what exactly probing is and what the interlock between client and server must be. This has led to some worry about the scalability of the probing process, and although the time required does scale linearly with the number of filesystems that the client may have state for with respect to a given server, the actual process can be done efficiently.

To address these issues we propose rewriting the above to be more clear and to give suggestions about how to do the required scanning efficiently.

5.4. Proposed changes to other sections

5.4.1. Proposed changes: callback update

Some changes are necessary to reduce confusion about the process of callback information update and in particular to make it clear that no state is freed as a result:

- Make it clear that after migration there are confirmed entries for transferred clientid4/nfs_client_id4 pairs.

- Be explicit in the sections headed "otherwise," in the descriptions of SETCLIENTID and SETCLIENTID_CONFIRM, that these don't apply in the cases we are concerned about.

5.4.2. Proposed changes: clientid4 handling

To address both of the clientid4-related issues mentioned in Section 4.5, we propose replacing the last three paragraphs of the section entitled "Client ID" with the following:

> Once a SETCLIENTID and SETCLIENTID_CONFIRM sequence has successfully completed, the client uses the shorthand client identifier, of type clientid4, instead of the longer and less compact nfs_client_id4 structure. This shorthand client identifier (a client ID) is assigned by the server and should be chosen so that it will not conflict with a client ID previously assigned by same server. This applies across server restarts or reboots.
>
> Distinct servers MAY assign clientid4's independently, and will generally do so. Therefore, a client has to be prepared to deal with multiple instances of the same clientid4 value received on distinct IP addresses, denoting separate entities. When trunking of server IP addresses is not a consideration, a client should keep track of (IP-address, clientid4) pairs, so that each pair is distinct. In the face of possible trunking of server IP addresses, the client will use the receipt of the same clientid4 from multiple IP-addresses, as an indication that the two IP-addresses may be trunked and proceed to determine, from the observed server behavior whether the two addresses are in fact trunked.
>
> When a clientid4 is presented to a server and that clientid4 is not recognized, the server will reject the request with the error NFS4ERR_STALE_CLIENTID. This can occur for a number of reasons:
>
> * A server reboot causing loss of the server's knowledge of the client
>
> * Client error sending an incorrect clientid4 or a valid clientid4 to the wrong server.
>
> * Loss of lease state due to lease expiration.

* Client or server error causing the server to believe that the client has rebooted (i.e. receiving a SETCLIENTID with an nfs_client_id4 which has a matching id string and a non-matching boot verifier).

* Migration of all state under the associated lease causes its non-existence to be recognized on the source server.

* Merger of state under the associated lease with another lease under a different clientid causes the clientid4 serving as the source of the merge to cease being recognized on its server.

In the event of a server reboot, or loss of lease state due to lease expiration, the client must obtain a new clientid4 by use of the SETCLIENTID operation and then proceed to any other necessary recovery for the server reboot case (See the section entitled "Server Failure and Recovery"). In cases of server or client error resulting in this error, use of SETCLIENTID to establish a new lease is desirable as well.

In the last two cases, different recovery procedures are required. Note that in cases in which there is any uncertainty about which sort of handling is applicable, the distinguishing characteristic is that in reboot-like cases, the clientid4 and all associated stateids cease to exist while in migration-related cases, the clientid4 ceases to exist while the stateids are still valid.

The client must also employ the SETCLIENTID operation when it receives a NFS4ERR_STALE_STATEID error using a stateid derived from its current clientid4, since this indicates a situation, such as server reboot which has invalidated the existing clientid4 and associated stateids (see the section entitled "lock-owner" for details).

See the detailed descriptions of SETCLIENTID and SETCLIENTID_CONFIRM for a complete specification of the operations.

5.4.3. Proposed changes: NFS4ERR_CLID_INUSE

It appears to be the intention that only a single principal be used for client establishment between any client-server pair. However:

o There is no explicit statement to this effect.

o The error that indicates a principal conflict has a name which does not clarify this issue: NFS4ERR_CLID_INUSE.

- The definition of the error is also not very helpful: "The SETCLIENTID operation has found that a client id is already in use by another client".

As a result, servers exist which reject a SETCLIENTID simply because there already exists a clientid for the same client, established using a different IP address. Although this is generally understood to be erroneous, such servers still exist and the spec should make the correct behavior clear.

Although the error name cannot be changed, the following changes should be made to avoid confusion:

- The definition of the error should be changed to read as follows:

  > The SETCLIENTID operation has found that the specified nfs_client_id4 was previously presented with a different principal and that client instance currently holds an active lease. A server MAY return this error if the same principal is used but a change in authentication flavor gives good reason to reject the new SETCLIENTID operation as not bona fide.

- In the description of SETCLIENTID, the phrase "then the server returns a NFS4ERR_CLID_INUSE error" should be expanded to read "then the server returns a NFS4ERR_CLID_INUSE error, since use of a single client with multiple principals is not allowed."

6. Results of proposed changes for NFSv4.0

The purpose of this section is to examine the troubling results reported in Section 3.1. We will look at the scenarios as they would be handled within the proposal.

Because the choice of uniform vs. non-uniform nfs_client_id4 id strings is a "SHOULD" in these cases, we will designate clients that follow this recommendation by SHOULD-UF-CID.

We will also have to take account of any merger-related "SHOULD" clauses to better understand how they have addressed the issues seen. We abbreviate as follows:

- SHOULD-SVR-AM refers to the server obeying the SHOULD which RECOMMENDS that they merge leases with identical nfs_client_id4 id strings and boot verifiers.

6.1. Results: Failure to free migrated state on client reboot

Let's look at the troublesome situation cited in Section 3.1.1. We have already seen what happens when SHOULD-UF-CID does not hold. Now let's look at the situation in which SHOULD-UF-CID holds, whether SHOULD-SVR-AM is in effect or not.

- A client C establishes a clientid4 C1 with server ABC specifying an nfs_client_id4 with id string value "C" and boot verifier 0x111.
- The client begins to access files in filesystem F on server ABC, resulting in generating stateids S1, S2, etc. under the lease for clientid C1. It may also access files on other filesystems on the same server.
- The filesystem is migrated from ABC to server XYZ. When transparent state migration is in effect, stateids S1 and S2 and lease {0x111, "C", C1} are now available for use by client C at server XYZ.
- Client C reboots and attempts to access data on server XYZ, whether in filesystem F or another. It does a SETCLIENTID with an nfs_client_id4 with id string value "C" and boot verifier 0x112. The state associated with lease {0x111, "C", C1} is deleted as part of creating {0x112, "C", C2}. No problem.

The correctness signature for this issue is

SHOULD-UF-CID

so if you have clients and servers that obey the SHOULD clauses, the problem is gone regardless of the choice on the MAY.

6.2. Results: Server reboots resulting in confused lease situation

Now let's consider the scenario given in Section 3.1.2. We have already seen what happens when SHOULD-UF-CID does not hold . Now let's look at the situation in which SHOULD-UF-CID holds and SHOULD-SVR-AM holds as well.

- Client C talks to server ABC using an nfs_client_id4 id string such as "C-ABC" and boot verifier v1. As a result a lease with clientid4 c.i established: {v1, "C-ABC", c.i}.
- Filesystem fs_a1 migrates from server ABC to server XYZ along with its state. Now server XYZ also has a lease: {v1, "C-ABC", c.i}

- Server ABC reboots.

- Client C talks to server ABC using an nfs_client_id4 id string such as "C-ABC" and boot verifier v1. As a result a lease with clientid4 c.j established: {v1, "C-ABC", c.j}.

- fs_a2 migrates from server ABC to server XYZ. As part of migration the incoming lease is seen to denote same nfs_client_id4 and so is merged with {v1, "C-ABC, c.i}.

- Now server XYZ has only one lease that matches {v1, "C_ABC", *}, so the problem is solved

Now let's consider the same scenario in the situation in which SHOULD-UF-CID holds and SHOULD-SVR-AM holds as well.

- Client C talks to server ABC using an nfs_client_id4 id string "C" and boot verifier v1. As a result a lease with clientid4 c.i is established: {v1, "C", c.i}.

- fs_a1 migrates from server ABC to server XYZ along with its state. Now XYZ also has a lease: {v1, "C", c.i}

- Server ABC reboots.

- Client C talks to server ABC using an nfs_client_id4 id string "C" and boot verifier v1. As a result a lease with clientid4 c.j is established: {v1, "C", c.j}.

- fs_a2 migrates from server ABC to server XYZ. As part of migration the incoming lease is seen to denote the same nfs_client_id4 and so is merged with {v1, "C", c.i}.

- Now server XYZ has only one lease that matches {v1, "C", *}, so the problem is solved

The correctness signature for this issue is

SHOULD-SVR-AM

so if you have clients and servers that obey the SHOULD clauses, the problem is gone regardless of the choice on the MAY.

6.3. Results: Client complexity issues

Consider the following situation:

- There are a set of clients C1 through Cn accessing servers S1 through Sm. Each server manages some significant number of filesystems with the filesystem count L being significantly greater than m.

- Each client Cx will access a subset of the servers and so will have up to m clientids, which we will call Cxy for server Sy.

- Now assume that for load-balancing or other operational reasons, numbers of filesystems are migrated among the servers. As a result, depending on how this handled, the number of clientids may explode. See below.

Now look what will happen under various scenarios:

- We have previously (in Section 3.1.3) looked at this in case of client following the non-uniform client-string approach. In that case, each client-server pair could have up to m clientids and each client will have up to m**2 clientids. If we add the possibility of server reboot, the only bound on a client's clientid count is L.

- If we look at this in the SHOULD-UF-CID case in which the SHOULD-SVR_AM condition holds, the situation is no different. Although the server has the client identity information that could enable same-client-same-server leases to be combined, it does not do so. We still have up to L clientids per client.

- On the other hand, if we look at the SHOULD-UF-CID case in which SHOULD-SVR-AM holds, the problem is gone. There can be no more than m clientids per client, and n clientids per server.

The correctness signature for this issue is

(SHOULD-UF-CID & SHOULD-SVR-AM)

so if you have clients and servers that obey the SHOULD clauses, the problem is gone regardless of the choice on the MAY.

6.4. Result summary

We have seen that (SHOULD-SVR-AM & SHOULD-UF-CID) are sufficient to solve the problems people have experienced.

7. Issues for NFSv4.1

Because NFSv4.1 embraces the uniform client-string approach, addressing migration issues is simpler. In the terms of Section 6, we already have SHOULD-UF-CID, for NFSv4.1, as advised by section 2.4 of [RFC5661], simplifying the work to be done.

Nevertheless, there are some issues that will have to be addressed. Some examples:

- The other necessary part of addressing migration issues, which we call above SHOULD-SVR-AM, is not currently addressed by NFSv4.1 and changes need to be made to make it clear that state needs to be appropriately merged as part of migration, to avoid multiple clientids between a client-server pair.

- There needs to be some clarification of how migration, and particularly transparent state migration, should interact with pNFS layouts.

- The current discussion (in [RFC5661]), of the possibility of server_owner changes is incomplete and confusing.

Discussion of how to resolve these issues will appear in the sections below.

7.1. Addressing state merger in NFSv4.1

The existing treatment of state transfer in [RFC5661], has similar problems to that in [RFC3530] and [RFC3530bis] in that it assumes that the state for multiple filesystems on different servers will not be merged to so that it appears under a single common clientid. We've already seen the reasons that this is a problem, with regard to NFSv4.0.

Although we don't have the problems stemming from the non-uniform client-string approach, there are a number of complexities in the existing treatment of state management in the section entitled "Lock State and File System Transitions" in [RFC5661] that make this non-trivial to address:

- Migration is currently treated together with other sorts of filesystem transitions including transitioning between replicas without any NFS4ERR_MOVED errors.

- There is separate handling and discussion of the cases of matching and non-matching server scopes.

- In the case of matching server scopes, the text calls for an impossible degree of transparency.

- In the case of non-matching server scopes, the text does not mention transparent state migration at all, resulting in a functional regression from NFSV4.0

7.2. Addressing pNFS relationship with migration

This is made difficult because, within the PNFS framework, migration might mean any of several things:

- Transfer of the MDS, leaving DS's alone.

  This would be minimally disruptive to those using layouts but would require the pNFS control protocol to support the DS being directed to a new MDS.

- Transfer of a DS, leaving everything else in place.

  Such a transfer can be handled without using migration at all. The server can recall/revoke layouts, as appropriate.

- Transfer of the filesystem to a new filesystem with both MDS and DS's moving.

  In such a transfer, an entirely different set of DS's will be at the target location. There may even be no pNFS support on the destination filesystem at all.

Migration needs to support both the first and last of these models.

7.3. Addressing server owner changes in NFSv4.1

Section 2.10.5 of [RFC5661] states the following.

> The client should be prepared for the possibility that eir_server_owner values may be different on subsequent EXCHANGE_ID requests made to the same network address, as a result of various sorts of reconfiguration events. When this happens and the changes result in the invalidation of previously valid forms of trunking, the client should cease to use those forms, either by dropping connections or by adding sessions. For a discussion of lock reclaim as it relates to such reconfiguration events, see Section 8.4.2.1.

While this paragraph is literally true in that such reconfiguration events can happen and clients have to deal with them, it is confusing

in that it can be read as suggesting that clients have to deal with them without disruption, which in general is impossible.

A clearer alternative would be:

> It is always possible that, as a result of various sorts of reconfiguration events, eir_server_scope and eir_server_owner values may be different on subsequent EXCHANGE_ID requests made to the same network address.
>
> In most cases such reconfiguration events will be disruptive and indicate that an IP address formerly connected to one server is now connected to an entirely different one.
>
> Some guidelines on client handling of such situations follow:
>
> * When eir_server_scope changes, the client has no assurance that any id's it obtained previously (e.g. file handles) can be validly used on the new server, and, even if the new server accepts them, there is no assurance that this is not due to accident. Thus it is best to treat all such state as lost/ stale although a client may assume that the probability of inadvertent acceptance is low and treat this situation as within the next case.
>
> * When eir_server_scope remains the same and eir_server_owner.so_major_id changes, the client can use filehandles it has and attempt reclaims. It may find that these are now stale but if NFS4ERR_STALE is not received, he can proceed to reclaim his opens.
>
> * When eir_server_scope and eir_server_owner.so_major_id remain the same, the client has to use the now-current values of eir_server-owner.so_minor_id in deciding on appropriate forms of trunking.

8. Security Considerations

The current definitive definitions of the NFSv4.0 protocol, [RFC3530] and [RFC3530bis] both agree. The section entitled "Security Considerations" encourages that clients protect the integrity of the SECINFO operation, any GETATTR operation for the fs_locations attribute, and the operations SETCLIENTID/SETCLIENTID_CONFIRM. A migration recovery event can use any or all of these operations. We do not recommend any change here.

9. IANA Considerations

This document does not require actions by IANA.

10. Acknowledgements

The editor and authors of this document gratefully acknowledge the contributions of Trond Myklebust of NetApp and Robert Thurlow of Oracle. We also thank Tom Haynes of NetApp and Spencer Shepler of Microsoft for their guidance and suggestions.

Special thanks go to members of the Oracle Solaris NFS team, especially Rick Mesta and James Wahlig, for their work implementing an NFSv4.0 migration prototype and identifying many of the issues documented here.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[RFC3530bis] Haynes, T., Ed. and D. Noveck, Ed., "Network File System (NFS) Version 4 Protocol", 2014, <http://www.ietf.org/id/draft-ietf-nfsv4-rfc3530bis-32.txt>.

Work in progress.

[RFC5661] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, January 2010.

11.2. Informative References

[NFS-ext] Noveck, D., "NFS Protocol Extension: Retrospect and Prospect", 2014, <http://www.ietf.org/id/draft-dnoveck-nfs-extension-01.txt>.

Work in progress.

[migr-v4.0-update]
Noveck, D., Ed., Shivam, P., Lever, C., and B. Baker,
"NFSv4.0 migration: Specification Update", 2013,
<http://www.ietf.org/id/
draft-ietf-nfsv4-rfc3530-migration-update-03.txt>.

Work in progress.

Authors' Addresses

David Noveck (editor)
26 Locust Avenue
Lexington, MA 02421
US

Phone: +1 781 572 8038
Email: david.noveck@emc.com

Piyush Shivam
Oracle Corporation
5300 Riata Park Ct.
Austin, TX 78727
US

Phone: +1 512 401 1019
Email: piyush.shivam@oracle.com

Charles Lever
Oracle Corporation
1015 Granger Avenue
Ann Arbor, MI 48104
US

Phone: +1 248 614 5091
Email: chuck.lever@oracle.com

Bill Baker
Oracle Corporation
5300 Riata Park Ct.
Austin, TX 78727
US

Phone: +1 512 401 1081
Email: bill.baker@oracle.com