Subject: Re: [PATCH] NFS: init client before declaration
Posted by Stanislav Kinsbursky on Wed, 23 May 2012 11:30:23 GMT
View Forum Message <> Reply to Message

On 23.05.2012 00:32, Myklebust, Trond wrote:
> On Tue, 2012-05-22 at 12:43 -0400, Trond Myklebust wrote:
>> On Tue, 2012-05-22 at 20:18 +0400, Stanislav Kinsbursky wrote:
>>> We have another problem here.
>>> nfs4_init_client() will try to create pipe dentries prior to set of NFS_CS_READY
>>> to the client. And dentries will be created since semaphore is dropped and
>>> per-net superblock variable is initialized already.
>>> But __rpc_pipefs_event() relays on the fact, that no dentries present.
>>> Looks like the problem was introduced by me in aad9487c...
>>> So maybe we should not call "continue" instead "__rpc_pipefs_event()", when
>>> client becomes ready?
>>> Looks like this will allow us to handle such races.
>>
>> Let me rework this patch a bit...
>
> The following is ugly, but it should be demonstrably correct, and will
> ensure that __rpc_pipefs_event() will only be called for fully
> initialised nfs_clients...
>
> 8< ---------- ---------
> From 90c3b9fe9faeae32c8f629e8b6cbf5f50bb9b295 Mon Sep 17 00:00:00 2001
> From: Trond Myklebust<Trond.Myklebust@netapp.com>
> Date: Tue, 22 May 2012 16:22:50 -0400
> Subject: [PATCH 1/2] NFSv4: Fix a race in the net namespace mount
>   notification
>
> Since the struct nfs_client gets added to the global nfs_client_list
> before it is initialised, it is possible that rpc_pipefs_event can
> end up trying to create idmapper entries on such a thing.
>
> The solution is to have the mount notification wait for the
> initialisation of each nfs_client to complete, and then to
> skip any entries for which the it failed.
>
> Reported-by: Stanislav Kinsbursky<skinsbursky@parallels.com>
> Signed-off-by: Trond Myklebust<Trond.Myklebust@netapp.com>
> ---
>   fs/nfs/client.c | 10 ++++++++++
>   fs/nfs/idmap.c | 15 +++++++++++++++
>   fs/nfs/internal.h | 1 +
>   3 files changed, 26 insertions(+), 0 deletions(-)
>
> diff --git a/fs/nfs/client.c b/fs/nfs/client.c
> index 60f7e4e..d3c8553 100644

```
> --- a/fs/nfs/client.c
> +++ b/fs/nfs/client.c
> @@ -583,6 +583,16 @@ found_client:
>    return clp;
>   }
>
> +static bool nfs_client_ready(const struct nfs_client *clp)
> +{
> + return clp->cl_cons_state<= NFS_CS_READY;
> +}
> +
> +int nfs_wait_client_ready(const struct nfs_client *clp)
> +{
> + return wait_event_killable(nfs_client_active_wq, nfs_client_ready(clp));
> +}
> +
>   /*
>    * Mark a server as ready or failed
>    */
> diff --git a/fs/nfs/idmap.c b/fs/nfs/idmap.c
> index 3e8edbe..c0753c5 100644
> --- a/fs/nfs/idmap.c
> +++ b/fs/nfs/idmap.c
> @@ -530,9 +530,24 @@ static struct nfs_client *nfs_get_client_for_event(struct net *net, int
event)
>    struct nfs_net *nn = net_generic(net, nfs_net_id);
>    struct dentry *cl_dentry;
>    struct nfs_client *clp;
> + int err;
>
> +restart:
>    spin_lock(&nn->nfs_client_lock);
>    list_for_each_entry(clp,&nn->nfs_client_list, cl_share_link) {
> +  /* Wait for initialisation to finish */
> +  if (clp->cl_cons_state>  NFS_CS_READY) {
> +   atomic_inc(&clp->cl_count);
> +   spin_unlock(&nn->nfs_client_lock);
> +   err = nfs_wait_client_ready(clp);
```

What about NFSv4.1 ?
It's clients NFS_CS_READY status depends on session establishing RPC calls...
Which in turn can hung up pipefs mount call...

Moreover, looks like pipefs dentries creation have to be synchronized by
nfs_client_lock somehow... Otherwise because of races we can get a client
without pipe dentry....

```
> +   nfs_put_client(clp);
> +   if (err)
> +    return NULL;
> +   goto restart;
> +  }
> +  /* Skip nfs_clients that failed to initialise */
> +  if (clp->cl_cons_state<  0)
> +   continue;
>     if (clp->rpc_ops !=&nfs_v4_clientops)
>      continue;
>     cl_dentry = clp->cl_idmap->idmap_pipe->dentry;
> diff --git a/fs/nfs/internal.h b/fs/nfs/internal.h
> index b777bda..3ee4040 100644
> --- a/fs/nfs/internal.h
> +++ b/fs/nfs/internal.h
> @@ -168,6 +168,7 @@ extern struct nfs_server *nfs_clone_server(struct nfs_server *,
>           struct nfs_fattr *,
>           rpc_authflavor_t);
>   extern void nfs_mark_client_ready(struct nfs_client *clp, int state);
> +extern int nfs_wait_client_ready(const struct nfs_client *clp);
>   extern int nfs4_check_client_ready(struct nfs_client *clp);
>   extern struct nfs_client *nfs4_set_ds_client(struct nfs_client* mds_clp,
>            const struct sockaddr *ds_addr,
```

--
Best regards,
Stanislav Kinsbursky