
Subject: [PATCH 2/4] NFS: replace per-net client lock by mutex
Posted by [Stanislav Kinsbursky](#) on Mon, 27 Feb 2012 13:49:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Lockdep is sad otherwise, because inode mutex is taken on PipeFS dentry creation, which can be called on mount notification, where this per-net client lock is taken on clients list walk.

Note: I used simple mutex instead of rw semaphore because of nfs_put_client->atomic_dec_and_mutex_lock() call. Probably, there is a better solution here.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

```
fs/nfs/client.c | 36 ++++++-----
fs/nfs/idmap.c | 4 ++-
fs/nfs/netns.h | 2 +-
3 files changed, 21 insertions(+), 21 deletions(-)
```

```
diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index 8563585..d15269f 100644
```

```
--- a/fs/nfs/client.c
```

```
+++ b/fs/nfs/client.c
```

```
@@ -72,9 +72,9 @@ static int nfs_get_cb_ident_idr(struct nfs_client *clp, int minorversion)
retry:
```

```
    if (!idr_pre_get(&nn->cb_ident_idr, GFP_KERNEL))
        return -ENOMEM;
- spin_lock(&nn->nfs_client_lock);
+ mutex_lock(&nn->nfs_client_lock);
    ret = idr_get_new(&nn->cb_ident_idr, clp, &clp->cl_cb_ident);
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);
    if (ret == -EAGAIN)
        goto retry;
    return ret;
```

```
@@ -321,10 +321,10 @@ void nfs_put_client(struct nfs_client *clp)
    dprintk("--> nfs_put_client({%d})\n", atomic_read(&clp->cl_count));
    nn = net_generic(clp->net, nfs_net_id);
```

```
- if (atomic_dec_and_lock(&clp->cl_count, &nn->nfs_client_lock)) {
+ if (atomic_dec_and_mutex_lock(&clp->cl_count, &nn->nfs_client_lock)) {
    list_del(&clp->cl_share_link);
    nfs_cb_idr_remove_locked(clp);
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);
```

```
    BUG_ON(!list_empty(&clp->cl_superblocks));
```

```

@@ -519,7 +519,7 @@ nfs_get_client(const struct nfs_client_initdata *cl_init,

/* see if the client already exists */
do {
- spin_lock(&nn->nfs_client_lock);
+ mutex_lock(&nn->nfs_client_lock);

    clp = nfs_match_client(cl_init);
    if (clp)
@@ -527,7 +527,7 @@ nfs_get_client(const struct nfs_client_initdata *cl_init,
    if (new)
        goto install_client;

- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);

    new = nfs_alloc_client(cl_init);
} while (!IS_ERR(new));
@@ -539,7 +539,7 @@ nfs_get_client(const struct nfs_client_initdata *cl_init,
install_client:
    clp = new;
    list_add(&clp->cl_share_link, &nn->nfs_client_list);
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);

    error = cl_init->rpc_ops->init_client(clp, timeparms, ip_addr,
        authflavour, noresvport);
@@ -554,7 +554,7 @@ install_client:
    * - make sure it's ready before returning
    */
found_client:
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);

    if (new)
        nfs_free_client(new);
@@ -1045,11 +1045,11 @@ static void nfs_server_insert_lists(struct nfs_server *server)
    struct nfs_client *clp = server->nfs_client;
    struct nfs_net *nn = net_generic(clp->net, nfs_net_id);

- spin_lock(&nn->nfs_client_lock);
+ mutex_lock(&nn->nfs_client_lock);
    list_add_tail_rcu(&server->client_link, &clp->cl_superblocks);
    list_add_tail(&server->master_link, &nn->nfs_volume_list);
    clear_bit(NFS_CS_STOP_RENEW, &clp->cl_res_state);
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);

```

```

}

@@ -1061,12 +1061,12 @@ static void nfs_server_remove_lists(struct nfs_server *server)
    if (clp == NULL)
        return;
    nn = net_generic(clp->net, nfs_net_id);
- spin_lock(&nn->nfs_client_lock);
+ mutex_lock(&nn->nfs_client_lock);
    list_del_rcu(&server->client_link);
    if (list_empty(&clp->cl_superblocks))
        set_bit(NFS_CS_STOP_RENEW, &clp->cl_res_state);
    list_del(&server->master_link);
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);

    synchronize_rcu();
}

@@ -1220,11 +1220,11 @@ nfs4_find_client_ident(struct net *net, int cb_ident)
    struct nfs_client *clp;
    struct nfs_net *nn = net_generic(net, nfs_net_id);

- spin_lock(&nn->nfs_client_lock);
+ mutex_lock(&nn->nfs_client_lock);
    clp = idr_find(&nn->cb_ident_idr, cb_ident);
    if (clp)
        atomic_inc(&clp->cl_count);
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);
    return clp;
}

@@ -1243,7 +1243,7 @@ nfs4_find_client_sessionid(struct net *net, const struct sockaddr *addr,
    struct nfs_client *clp;
    struct nfs_net *nn = net_generic(net, nfs_net_id);

- spin_lock(&nn->nfs_client_lock);
+ mutex_lock(&nn->nfs_client_lock);
    list_for_each_entry(clp, &nn->nfs_client_list, cl_share_link) {
        if (nfs4_cb_match_client(addr, clp, 1) == false)
            continue;
@@ -1257,10 +1257,10 @@ nfs4_find_client_sessionid(struct net *net, const struct sockaddr
*addr,
    continue;

    atomic_inc(&clp->cl_count);
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);

```

```

    return clp;
}
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);
    return NULL;
}

@@ -1781,7 +1781,7 @@ void nfs_clients_init(struct net *net)
#ifdef CONFIG_NFS_V4
    idr_init(&nn->cb_ident_idr);
#endif
- spin_lock_init(&nn->nfs_client_lock);
+ mutex_init(&nn->nfs_client_lock);
}

#ifdef CONFIG_PROC_FS
diff --git a/fs/nfs/idmap.c b/fs/nfs/idmap.c
index b5c6d8e..98b0b6b 100644
--- a/fs/nfs/idmap.c
+++ b/fs/nfs/idmap.c
@@ -561,7 +561,7 @@ static int rpc_pipefs_event(struct notifier_block *nb, unsigned long event,
    struct nfs_client *clp;
    int error = 0;

- spin_lock(&nn->nfs_client_lock);
+ mutex_lock(&nn->nfs_client_lock);
    list_for_each_entry(clp, &nn->nfs_client_list, cl_share_link) {
        if (clp->rpc_ops != &nfs_v4_clientops)
            continue;
@@ -569,7 +569,7 @@ static int rpc_pipefs_event(struct notifier_block *nb, unsigned long event,
        if (error)
            break;
    }
- spin_unlock(&nn->nfs_client_lock);
+ mutex_unlock(&nn->nfs_client_lock);
    return error;
}

diff --git a/fs/nfs/netns.h b/fs/nfs/netns.h
index 7baad89..24d0bc3 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -12,7 +12,7 @@ struct nfs_net {
#ifdef CONFIG_NFS_V4
    struct idr cb_ident_idr; /* Protected by nfs_client_lock */
#endif
- spinlock_t nfs_client_lock;
+ struct mutex nfs_client_lock;

```

};

extern int nfs_net_id;

Subject: Re: [PATCH 2/4] NFS: replace per-net client lock by mutex

Posted by [Myklebust, Trond](#) on Mon, 27 Feb 2012 15:00:18 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Mon, 2012-02-27 at 17:49 +0400, Stanislav Kinsbursky wrote:

> Lockdep is sad otherwise, because inode mutex is taken on PipeFS dentry
> creation, which can be called on mount notification, where this per-net client
> lock is taken on clients list walk.

>

> Note: I used simple mutex instead of rw semaphore because of
> nfs_put_client->atomic_dec_and_mutex_lock() call. Probably, there is a better
> solution here.

>

> Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>

>

This is overkill... We end up blocking NFSv4 callbacks while the
rpc_pipefs notifier runs through the nfs_clients creating or destroying
idmapper dentries.

Surely the rpc_pipefs_event() can take a reference to the nfs_client and
then drop the spin_lock if it sees that it needs to create or destroy a
dentry?

--

Trond Myklebust

Linux NFS client maintainer

NetApp

Trond.Myklebust@netapp.com

www.netapp.com

Subject: Re: [PATCH 2/4] NFS: replace per-net client lock by mutex

Posted by [Stanislav Kinsbursky](#) on Mon, 27 Feb 2012 15:42:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

> On Mon, 2012-02-27 at 17:49 +0400, Stanislav Kinsbursky wrote:

>> Lockdep is sad otherwise, because inode mutex is taken on PipeFS dentry
>> creation, which can be called on mount notification, where this per-net client
>> lock is taken on clients list walk.

>>
>> Note: I used simple mutex instead of rw semaphore because of
>> nfs_put_client->atomic_dec_and_mutex_lock() call. Probably, there is a better
>> solution here.
>>
>> Signed-off-by: Stanislav Kinsbursky<skinsbursky@parallels.com>
>>
>
> This is overkill... We end up blocking NFSv4 callbacks while the
> rpc_pipefs notifier runs through the nfs_clients creating or destroying
> idmapper dentries.
>
> Surely the rpc_pipefs_event() can take a reference to the nfs_client and
> then drop the spin_lock if it sees that it needs to create or destroy a
> dentry?
>

Sure, thanks for notice.
Looks like this logic also works to SUNRPC clients.
I'll send updated version soon.

--
Best regards,
Stanislav Kinsbursky
