Subject: [PATCH 4/4] NFS: make nfs_client_lock per net ns
Posted by Stanislav Kinsbursky on Mon, 23 Jan 2012 17:26:31 GMT
View Forum Message <> Reply to Message

This patch makes nfs_clients_lock allocated per network namespace. All items it
protects are already network namespace aware.

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

---
 fs/nfs/client.c | 51 ++++++++++++++++++++++++++++++++----------------
 fs/nfs/idmap.c | 4 ++--
 fs/nfs/internal.h | 3 ---
 fs/nfs/netns.h | 1 +
 4 files changed, 32 insertions(+), 27 deletions(-)

diff --git a/fs/nfs/client.c b/fs/nfs/client.c
index f51b279..9e11d29 100644
--- a/fs/nfs/client.c
+++ b/fs/nfs/client.c
@@ -55,7 +55,6 @@

 #define NFSDBG_FACILITY NFSDBG_CLIENT

-DEFINE_SPINLOCK(nfs_client_lock);
 static DECLARE_WAIT_QUEUE_HEAD(nfs_client_active_wq);
 #ifdef CONFIG_NFS_V4

@@ -73,9 +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(&nfs_client_lock);
+ spin_lock(&nn->nfs_client_lock);
  ret = idr_get_new(&nn->cb_ident_idr, clp, &clp->cl_cb_ident);
- spin_unlock(&nfs_client_lock);
+ spin_unlock(&nn->nfs_client_lock);
  if (ret == -EAGAIN)
   goto retry;
  return ret;
@@ -313,15 +312,18 @@ static void nfs_free_client(struct nfs_client *clp)
  */
 void nfs_put_client(struct nfs_client *clp)
 {
+ struct nfs_net *nn;
+
  if (!clp)
   return;

+ nn = net_generic(clp->net, nfs_net_id);
  dprintk("--> nfs_put_client({%d})\n", atomic_read(&clp->cl_count));

- if (atomic_dec_and_lock(&clp->cl_count, &nfs_client_lock)) {
+ if (atomic_dec_and_lock(&clp->cl_count, &nn->nfs_client_lock)) {
   list_del(&clp->cl_share_link);
   list_del_init(&clp->cl_superblocks);
- spin_unlock(&nfs_client_lock);
+ spin_unlock(&nn->nfs_client_lock);

   BUG_ON(!clp->cl_cons_state ||
          (clp->cl_cons_state != -ENOMEM &&

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

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

@@ -516,7 +518,7 @@ nfs_get_client(const struct nfs_client_initdata *cl_init,

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

    clp = nfs_match_client(cl_init);
    if (clp)
@@ -524,7 +526,7 @@ nfs_get_client(const struct nfs_client_initdata *cl_init,
    if (new)
     goto install_client;

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

    new = nfs_alloc_client(cl_init);
   } while (!IS_ERR(new));
@@ -536,7 +538,7 @@ nfs_get_client(const struct nfs_client_initdata *cl_init,
   clp = new;
   list_add(&clp->cl_share_link, &nn->nfs_client_list);
- spin_unlock(&nfs_client_lock);
+ spin_unlock(&nn->nfs_client_lock);

   error = cl_init->rpc_ops->init_client(clp, timeparms, ip_addr,
           authflavour, noresvport);
@@ -551,7 +553,7 @@ install_client:
    * - make sure it's ready before returning
    */
 found_client:
- spin_unlock(&nfs_client_lock);
+ spin_unlock(&nn->nfs_client_lock);

   if (new)
```

```
   nfs_free_client(new);
@@ -1041,24 +1043,25 @@ 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(&nfs_client_lock);
+ spin_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(&nfs_client_lock);
+ spin_unlock(&nn->nfs_client_lock);

 }

 static void nfs_server_remove_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(&nfs_client_lock);
+ spin_lock(&nn->nfs_client_lock);
  list_del_rcu(&server->client_link);
  if (clp && list_empty(&clp->cl_superblocks))
   set_bit(NFS_CS_STOP_RENEW, &clp->cl_res_state);
  list_del(&server->master_link);
- spin_unlock(&nfs_client_lock);
+ spin_unlock(&nn->nfs_client_lock);

  synchronize_rcu();
 }
@@ -1212,11 +1215,11 @@ 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(&nfs_client_lock);
+ spin_lock(&nn->nfs_client_lock);
  clp = idr_find(&nn->cb_ident_idr, cb_ident);
  if (clp)
   atomic_inc(&clp->cl_count);
- spin_unlock(&nfs_client_lock);
+ spin_unlock(&nn->nfs_client_lock);
  return clp;
 }

@@ -1235,7 +1238,7 @@ nfs4_find_client_sessionid(const struct sockaddr *addr,
  struct nfs_client *clp;
  struct nfs_net *nn = net_generic(&init_net, nfs_net_id);
```

```
- spin_lock(&nfs_client_lock);
+ spin_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;
@@ -1249,10 +1252,10 @@ nfs4_find_client_sessionid(const struct sockaddr *addr,
    continue;

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

@@ -1849,7 +1852,7 @@ static void *nfs_server_list_start(struct seq_file *m, loff_t *_pos)
  struct nfs_net *nn = net_generic(m->private, nfs_net_id);

  /* lock the list against modification */
- spin_lock(&nfs_client_lock);
+ spin_lock(&nn->nfs_client_lock);
  return seq_list_start_head(&nn->nfs_client_list, *_pos);
 }

@@ -1868,7 +1871,9 @@ static void *nfs_server_list_next(struct seq_file *p, void *v, loff_t *pos)
  */
 static void nfs_server_list_stop(struct seq_file *p, void *v)
 {
- spin_unlock(&nfs_client_lock);
+ struct nfs_net *nn = net_generic(p->private, nfs_net_id);
+
+ spin_unlock(&nn->nfs_client_lock);
 }

 /*
@@ -1930,7 +1935,7 @@ static void *nfs_volume_list_start(struct seq_file *m, loff_t *_pos)
  struct nfs_net *nn = net_generic(m->private, nfs_net_id);

  /* lock the list against modification */
- spin_lock(&nfs_client_lock);
+ spin_lock(&nn->nfs_client_lock);
  return seq_list_start_head(&nn->nfs_volume_list, *_pos);
 }
```

```
@@ -1949,7 +1954,9 @@ static void *nfs_volume_list_next(struct seq_file *p, void *v, loff_t *pos)
  */
 static void nfs_volume_list_stop(struct seq_file *p, void *v)
 {
- spin_unlock(&nfs_client_lock);
+ struct nfs_net *nn = net_generic(p->private, nfs_net_id);
+
+ spin_unlock(&nn->nfs_client_lock);
 }

 /*
diff --git a/fs/nfs/idmap.c b/fs/nfs/idmap.c
index 92deaf8..aed3d2e 100644
--- a/fs/nfs/idmap.c
+++ b/fs/nfs/idmap.c
@@ -575,7 +575,7 @@ static int rpc_pipefs_event(struct notifier_block *nb, unsigned long event,
  struct nfs_client *clp;
  int error = 0;

- spin_lock(&nfs_client_lock);
+ spin_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;
@@ -583,7 +583,7 @@ static int rpc_pipefs_event(struct notifier_block *nb, unsigned long event,
   if (error)
    break;
  }
- spin_unlock(&nfs_client_lock);
+ spin_unlock(&nn->nfs_client_lock);
  return error;
 }

diff --git a/fs/nfs/internal.h b/fs/nfs/internal.h
index 958fff2..b38b733 100644
--- a/fs/nfs/internal.h
+++ b/fs/nfs/internal.h
@@ -182,9 +182,6 @@ static inline void nfs_fs_proc_exit(void)
 {
 }
 #endif
-#ifdef CONFIG_NFS_V4
-extern spinlock_t nfs_client_lock;
-#endif

 /* nfs4namespace.c */
 #ifdef CONFIG_NFS_V4
diff --git a/fs/nfs/netns.h b/fs/nfs/netns.h
```

```
index 547cc95..7baad89 100644
--- a/fs/nfs/netns.h
+++ b/fs/nfs/netns.h
@@ -12,6 +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;
 };

 extern int nfs_net_id;
```