Subject: [PATCH v3 2/2] SUNRPC: move per-net operations from svc_destroy()
Posted by Stanislav Kinsbursky on Wed, 02 May 2012 12:08:45 GMT
View Forum Message <> Reply to Message

v2:
1) Increase per-net usage counted in lockd_up_net(), because of note 2.

The idea is to separate service destruction and per-net operations, because
these are two different things and it's mix looks ugly.

Notes:
1) For NFS server this patch looks ugly (sorry for that). But these place will be
rewritten soon during NFSd containerization.
2) LockD per-net counter increase int lockd_up() was moved prior to
make_socks() to make lockd_down_net() call safe in case of error.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 fs/lockd/svc.c | 27 +++++++++++++++++++----------
 fs/nfs/callback.c | 3 +++
 fs/nfsd/nfsctl.c | 4 ++++
 fs/nfsd/nfssvc.c | 7 +++++++
 net/sunrpc/svc.c | 4 ----
 5 files changed, 29 insertions(+), 16 deletions(-)

diff --git a/fs/lockd/svc.c b/fs/lockd/svc.c
index b7e92ed..3250f28 100644
--- a/fs/lockd/svc.c
+++ b/fs/lockd/svc.c
@@ -257,7 +257,7 @@ static int lockd_up_net(struct net *net)
  struct svc_serv *serv = nlmsvc_rqst->rq_server;
  int error;

- if (ln->nlmsvc_users)
+ if (ln->nlmsvc_users++)
   return 0;

  error = svc_rpcb_setup(serv, net);
@@ -272,6 +272,7 @@ static int lockd_up_net(struct net *net)
  svc_rpcb_cleanup(serv, net);
 err_rpcb:
+ ln->nlmsvc_users--;
  return error;
 }

@@ -299,6 +300,7 @@ int lockd_up(struct net *net)
 {

```
  struct svc_serv *serv;
  int  error = 0;
+ struct lockd_net *ln = net_generic(net, lockd_net_id);

  mutex_lock(&nlmsvc_mutex);
  /*
@@ -330,9 +332,11 @@ int lockd_up(struct net *net)
   goto destroy_and_out;
  }

+ ln->nlmsvc_users++;
+
  error = make_socks(serv, net);
  if (error < 0)
-  goto destroy_and_out;
+  goto err_start;

  /*
   * Create the kernel thread and wait for it to start.
@@ -344,7 +348,7 @@ int lockd_up(struct net *net)
    printk(KERN_WARNING
     "lockd_up: svc_rqst allocation failed, error=%d\n",
     error);
-  goto destroy_and_out;
+  goto err_start;
  }

  svc_sock_update_bufs(serv);
@@ -358,7 +362,7 @@ int lockd_up(struct net *net)
   nlmsvc_rqst = NULL;
   printk(KERN_WARNING
    "lockd_up: kthread_run failed, error=%d\n", error);
-  goto destroy_and_out;
+  goto err_start;
  }

  /*
@@ -368,14 +372,14 @@ int lockd_up(struct net *net)
 destroy_and_out:
  svc_destroy(serv);
 out:
- if (!error) {
-  struct lockd_net *ln = net_generic(net, lockd_net_id);
-
-  ln->nlmsvc_users++;
+ if (!error)
   nlmsvc_users++;
- }
```

```
  mutex_unlock(&nlmsvc_mutex);
  return error;
+
+err_start:
+ lockd_down_net(net);
+ goto destroy_and_out;
 }
 EXPORT_SYMBOL_GPL(lockd_up);

@@ -386,11 +390,10 @@ void
 lockd_down(struct net *net)
 {
  mutex_lock(&nlmsvc_mutex);
+ lockd_down_net(net);
  if (nlmsvc_users) {
-  if (--nlmsvc_users) {
-   lockd_down_net(net);
+  if (--nlmsvc_users)
    goto out;
-  }
  } else {
   printk(KERN_ERR "lockd_down: no users! task=%p\n",
    nlmsvc_task);
diff --git a/fs/nfs/callback.c b/fs/nfs/callback.c
index 26b38fb..cff3940 100644
--- a/fs/nfs/callback.c
+++ b/fs/nfs/callback.c
@@ -314,6 +314,8 @@ out_err:
  dprintk("NFS: Couldn't create callback socket or server thread; "
   "err = %d\n", ret);
  cb_info->users--;
+ if (serv)
+  svc_shutdown_net(serv, net);
  goto out;
 }

@@ -328,6 +330,7 @@ void nfs_callback_down(int minorversion)
  cb_info->users--;
  if (cb_info->users == 0 && cb_info->task != NULL) {
   kthread_stop(cb_info->task);
+  svc_shutdown_net(cb_info->serv, current->nsproxy->net_ns);
   svc_exit_thread(cb_info->rqst);
   cb_info->serv = NULL;
   cb_info->rqst = NULL;
diff --git a/fs/nfsd/nfsctl.c b/fs/nfsd/nfsctl.c
index 7269988..efb3818 100644
--- a/fs/nfsd/nfsctl.c
+++ b/fs/nfsd/nfsctl.c
```

```
@@ -672,6 +672,8 @@ static ssize_t __write_ports_addfd(char *buf)

  err = svc_addsock(nfsd_serv, fd, buf, SIMPLE_TRANSACTION_LIMIT);
  if (err < 0) {
+  if (nfsd_serv->sv_nrthreads == 1)
+   svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
   svc_destroy(nfsd_serv);
   return err;
  }
@@ -740,6 +742,8 @@ out_close:
   svc_xprt_put(xprt);
  }
 out_err:
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
  svc_destroy(nfsd_serv);
  return err;
 }
diff --git a/fs/nfsd/nfssvc.c b/fs/nfsd/nfssvc.c
index 0762f3c..3fffe6c 100644
--- a/fs/nfsd/nfssvc.c
+++ b/fs/nfsd/nfssvc.c
@@ -426,6 +426,9 @@ int nfsd_set_nrthreads(int n, int *nthreads)
   if (err)
    break;
  }
+
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
  svc_destroy(nfsd_serv);

  return err;
@@ -473,6 +476,8 @@ out_shutdown:
  if (error < 0 && !nfsd_up_before)
   nfsd_shutdown();
 out_destroy:
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
  svc_destroy(nfsd_serv);  /* Release server */
 out:
  mutex_unlock(&nfsd_mutex);
@@ -670,6 +675,8 @@ int nfsd_pool_stats_release(struct inode *inode, struct file *file)
  int ret = seq_release(inode, file);
  mutex_lock(&nfsd_mutex);
  /* this function really, really should have been called svc_put() */
+ if (nfsd_serv->sv_nrthreads == 1)
+  svc_shutdown_net(nfsd_serv, current->nsproxy->net_ns);
  svc_destroy(nfsd_serv);
```

```
  mutex_unlock(&nfsd_mutex);
  return ret;
diff --git a/net/sunrpc/svc.c b/net/sunrpc/svc.c
index e6d542c..b7210f5 100644
--- a/net/sunrpc/svc.c
+++ b/net/sunrpc/svc.c
@@ -537,8 +537,6 @@ EXPORT_SYMBOL_GPL(svc_shutdown_net);
 void
 svc_destroy(struct svc_serv *serv)
 {
- struct net *net = current->nsproxy->net_ns;
-
  dprintk("svc: svc_destroy(%s, %d)\n",
     serv->sv_program->pg_name,
     serv->sv_nrthreads);
@@ -553,8 +551,6 @@ svc_destroy(struct svc_serv *serv)

  del_timer_sync(&serv->sv_temptimer);

- svc_shutdown_net(serv, net);
-
  /*
   * The last user is gone and thus all sockets have to be destroyed to
   * the point. Check this.
```